Robust and Large Scale Network Optimization in Logistics

Von der
Carl-Friedrich-Gauß-Fakultät
der Technischen Universität Carolo-Wilhelmina zu Braunschweig

zur Erlangung des Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigte Dissertation

von
Dipl. Math. Alexander T. Richter
geboren am 15. April 1985
in Berlin

Eingereicht am:	3. März 2017
Disputation am:	4. Mai 2017
1. Referentin/Referent:	Prof. Dr. Sebastian Stiller
2. Referentin/Referent:	Prof. Dr. Britta Peis

2018

Copyright:
© 2018 Alexander T. Richter

Acknowledgements

This research has evolved from very exciting and inspiring years working at the Combinatorial Optimization and Graph Algorithms group at TU Berlin and the Discrete Optimization group at TU Braunschweig. Writing this thesis would however not have been possible without the guidance and support of many people whom I wish to thank in the following. First and foremost I am deeply indebted to Sebastian Stiller for his generous support, encouragement and the supervision of my thesis. He introduced me to the exciting field of Robust Optimization and his enthusiasm for its application to logistics networks was more than intriguing. I am also grateful to Britta Peis who not only kindly agreed to referee this thesis but also accompanied my first steps towards more theoretic topics.

Financially, this work has been supported by the European Regional Development Fund as part of the research projects *MultiTrans* and *RobuNet* on Optimization for Logistics Networks and I wish to thank Prof. Dr. Rolf Möhring and Prof. Dr. Martin Skutella for their introduction into the fields of Combinatorial Optimization and for having established such a fruitful research environment where theory meets practice.

A considerable part of this work arose from these projects and I enjoyed discussions, code reviews, and the ups and downs with my former project members Wiebke Höhn, Pritta Peis, Yann Disser, Tobias Harks, Falk Hüffner, Daniel Karch, Felix König, Jannik Matuschke, Jens Schulz, and Sebastian Stiller. In this regard also, a special thanks goes to Jannik and Felix for my project onboarding from the very beginning and to Wiebke for providing structure when it was most needed. Both projects were in cooperation with 4flow AG and I am grateful to Christina Hayden, Barbara Schenk, Wendelin Groß, and Lars Stolletz for facilitating to work with and do evaluations on real world case studies and for fruitful discussions on the subjects.

Besides the work on the projects I also enjoyed working on theoretic aspects with my coauthors Nicole Megow, Julie Meißner, Fidaa Abed, Lin Chen, Yann Disser, Martin Groß, and Roman Rischke. Since especially the last chapters of a thesis require an increased devotion I am very grateful to have found in Braunschweig a calm and productive working environment, which I happily shared with my esteemed colleagues Imke Joormann and Christoph Hansknecht.

Moreover, I would like to thank Miriam Schlöter, Julie Meißner, Karl Däubel, Torsten Gellert, Kai-Simon Goetzmann, Jan Hackfeld, and Frieder Smolny for carefully proofreading parts of this thesis.

Finally, I would like to thank my family for their unconditional support throughout my whole life and Katja for her belief in me and the many happy moments we share.

Berlin, April 2018 *Alexander T. Richter*

Abstract

We consider combinatorial optimization problems for network flows, a natural model for the transportation of commodities. Network design models build upon network flow models and capture the tradeoff between investing in the network structure and benefits for the resulting network flow such as cost savings or maximum throughput.

Classical network optimization models have received considerable attention throughout the last decades, yet they typically focus on optimization aspects that are considered central for real world problems. However, the real world is rarely simple or easy to optimize. This naturally calls for more accurate models that are closer to reality and this thesis explores possibilities and limitations of extending classical network flow models. We propose new mathematical models for transport optimization in logistics networks that feature commodities with multidimensional properties, e.g. their mass and volume, to capture consolidation effects of commodities with complementing properties. We provide both new theoretical insights as well as solution methods with immediate practical impact.

The first model is for transportation planning on the tactical level and incorporates temporal consolidation effects via a cyclic pattern expansion of the network. To solve the model, we propose various heuristics, including a local search procedure that re-routes flow of multiple commodities at once. We complement our heuristics by lower bounds from an aggregated mixed integer programming formulation with strengthened inequalities. In a case study from the automotive, chemical, and retail industry, we prove that most of our solutions are within a single-digit percentage of the optimum.

To provide for a theoretical background, we consider problem variants where each commodity is required to be routed unsplittably. We focus on consolidation effects and prove hardness results for various special cases. For more general cases, we introduce the notion of forest costs, which overestimate real costs, and propose a dynamic program that finds solutions with optimal forest costs.

We also devise a model for strategic route planning under uncertainty and provide for a robust optimization method that anticipates the fluctuation of demands by minimizing the worst-case cost over a restricted scenario set. We show that the corresponding adversary problem is $\mathcal{NP}$-hard. To still find solutions with very good worst-case cost, we derive a carefully relaxed and simplified mixed integer linear program, which solves well for large instances due to its strong linear programming relaxation. The results for real-world instances show that robust optimization can significantly reduce worst-case cost.

Moreover, this strategic model can easily be extended to include hub decisions. We present an algorithmic toolkit to obtain robust solutions for a M-median hub location problem for logistics networks. We address two concepts for robustness: Demand robustness as before and incremental hub chains. The idea behind incremental hub chains is to provide solutions

for every number of hubs to operate, such that they are robust under changes of this number. We propose a flexible, decremental framework that can be applied to large instances when combined with fast combinatorial heuristics. To assess the price of being incremental we compare incremental solutions with M-median solutions that are obtained with an LP-based hub search on two medium sized instances.

For demand robustness we apply the LP-based hub search to these instances for various uncertainty sets derived from historical data. For our instances we find a price of robustness that is moderate for interval uncertainty sets that use average demand values as lower bounds. Trend based uncertainty sets, which are obtained with an outlier method, show a considerable tradeoff between historical average costs and worst case costs.

Finally, we investigate the problem of scheduling the maintenance of edges in a network, motivated by the goal of minimizing outages in transportation or telecommunication networks. We focus on maintaining connectivity between two nodes over time; for the special case of path networks, this is related to the problem of minimizing the busy time of machines. We show that the problem can be solved in polynomial time in arbitrary networks if preemption is allowed. If preemption is restricted to integral time points, the problem is NP-hard and for the non-preemptive case, we show strong non-approximability results. Furthermore, we give tight bounds on the power of preemption, i.e., the maximum ratio of the values of non-preemptive and preemptive optimal solutions.

Zusammenfassung

Wir betrachten kombinatorische Optimierungsprobleme für Netzwerkflüsse, ein natürliches Modell für den Transport von mehreren Gütern. Netzwerkdesignmodelle bauen auf Netzwerkflussmodellen auf. Sie suchen einen Kompromiss, um einerseits die Kosten für Investitionen in die Netzwerkstruktur zu minimieren und andererseits den resultierenden Nutzen für dem Netzwerk zugrunde liegende Flüsse zu maximieren, etwa Kostenersparnisse oder höheren Wert des Flusses.

Klassische Netzwerkoptimierungsprobleme wurden in den letzten Jahrzehnten eingehend in der Literatur betrachtet. Dennoch konzentrieren sie sich meist auf Optimierungsaspekte, welche als zentral für die Problemstellung angesehen werden. Allerdings sind reale Modelle meist weder einfach noch leicht zu optimieren. Dies verlangt auf natürliche Weise nach genaueren Modellen, die näher an der Realität sind. Diese Doktorarbeit untersucht Möglichkeiten und Grenzen, klassische Netzwerkprobleme zu erweitern.

Wir schlagen neue mathematische Modelle für die Transportoptimierung in der Logistik vor, welche sich durch die Berücksichtigung von mehrdimensionale Eigenschaften der Güter auszeichnen, wie zum Beispiel Masse oder Volumen. Dies erlaubt es, Konsolidierungseffekte von Gütern abzubilden, welche gegensätzliche Eigenschaften aufweisen. Wir erzielen sowohl neue theoretische Einsichten als auch Lösungsmethoden von praktischer Relevanz.

Das erste Modell ist für die Transportplanung auf der taktischen Ebene und bezieht zeitliche Konsolidierungseffekte durch eine zyklische Zeitexpansion mit ein. Um dieses Modell zu lösen, schlagen wir verschiedene Heuristiken vor, darunter eine Lokale Suche, welche mehrere Güter gleichzeitig umverteilt. Den Heuristiken stehen untere Schranken von einem aggregierten, gemischt ganzzahligen Programm mit gestärkten Ungleichungen gegenüber. In einer Fallstudie mit Instanzen aus der Automobilindustrie, der Chemiebranche und aus dem Einzelhandel zeigen wir auf, dass die meisten unserer Lösungen eine Optimalitätslücke von weniger als 10% aufweisen.

Um den theoretischen Hintergrund zu untersuchen, betrachten wir Problemstellungen, in denen Güter unteilbar durch das Netzwerk transportiert werden. Wir konzentrieren uns auf Konsolidierungseffekte und zeigen Härteresultate für verschiedene Spezialfälle. Für allgemeinere Fälle führen wir das Konzept von Baumkosten ein, welche die echten Kosten im Allgemeinen überschätzen, und stellen ein dynamische Programm vor, welches Lösungen mit optimalen Baumkosten berechnet.

Wir erarbeiten außerdem ein Modell für strategische Routenplanung unter Unsicherheit sowie eine robuste Optimierungsmethode, welche Nachfrageschwankungen antizipiert, indem Worst-Case-Kosten über einer beschränkten Szenarienmenge minimiert werden. Wir zeigen, dass das resultierende Gegenspielerproblem $\mathcal{NP}$-schwer ist. Um dennoch Lösungen mit sehr guten Worst-Case-Kosten zu finden, leiten wir ein sorgfältig relaxiertes und vereinfachtes

gemischt ganzzahliges Programm ab, welches sich aufgrund seiner starken LP-Relaxierung gut für große Instanzen lösen lässt. Ergebnisse auf realistischen Instanzen zeigen, dass robuste Optimierung die Worst-Case-Kosten deutlich senken kann.

Dieses Modell kann darüber hinaus leicht erweitert werden, um Hubentscheidungen zu berücksichtigen. Wir stellen einen algorithmischen Werkzeugkasten vor, um robuste Lösungen für ein M-Median Hub Location Problem für Logistiknetzwerke zu berechnen. Dabei befassen wir uns mit zwei Robustheitskonzepten: Robustheit gegen Nachfrageschwankungen wie zuvor und inkrementale Hubketten. Die Idee von inkrementalen Hubketten besteht darin, Lösungen für jede mögliche Anzahl an Hubstandorten anzugeben, sodass sie gegen Änderungen dieser Anzahl robust sind. Unser dekrementaler algorithmischer Rahmen kann auf sehr große Netzwerke angewendet werden, wenn er mit schnellen kombinatorischen Heuristiken kombiniert wird. Wir beurteilen den Preis der Inkrementalität, also die Mehrkosten einer Lösung dafür, dass sie zu einer inkrementalen Hubkette gehört, anhand von zwei mittelgroßen Instanzen. Wir vergleichen Glieder einer inkrementalen Hubkette mit entsprechenden M-Median Lösungen die wir mit einer LP-basierten Hubsuche erhalten.

Für das Konzept der Robustheit gegen Nachfrageschwankungen wenden wir ebenfalls die LP-basierten Hubsuche auf diese Instanzen an und leiten verschiedene Schwankungsintervallen von historischen Daten ab. Bei Intervallen, die eine Durchschnittsnachfragemenge als untere Intervallgrenze verwenden, beobachten wir einen moderaten Preis der Robustheit. Bei Trend basierten Schwankungsintervalle, welche wir mit einer Outliermethode erzeugen, stellen wir einen deutlichen Trade-off zwischen historischen Durchschnittskosten und Worst-Case-Kosten fest.

Zuletzt untersuchen wir die Problemstellung der Planung von Wartungsarbeiten an Kanten in einem Netzwerk, welche durch die Zielsetzung motiviert wird, Ausfälle in Transport- oder Telekommunikationsnetzwerken zu minimieren. Wir konzentrieren uns darauf, die Konnektivität zwischen zwei Knoten über die Zeit aufrecht zu erhalten. Für den Spezialfall eines Pfades ist dies verwandt mit dem Problem, die Belegungszeit von Maschinen zu minimieren.

Wir zeigen, dass das Problem mit polynomieller Zeitkomplexität in beliebigen Netzwerken gelöst werden kann, falls Wartungsarbeiten unterbrochen werden dürfen. Falls dies nur zu ganzzahligen Zeitpunkten erlaubt ist, ist das Problem bereits $\mathcal{NP}$-schwer. Für den Fall ohne Unterbrechungen zeigen wir starke Nichtapproximierbarkeitsresultate. Außerdem bestimmen wir scharfe Schranken für die Ersparnis durch Unterbrechungen.

Contents

Chapter 1

Introduction

The ongoing globalization of markets over the past decades accounts for an ever-increasing shipping volume of goods worldwide. In all industries, companies operate facilities spread out across the world to maximize profitability, and procurement and distribution have become global operations. The ensuing demand for transportation has fostered the growth of huge international logistics networks with the potential to increase efficiency through economies of scale, pooling of orders, and a global view on network layout.

Facilities or objects, representing the nodes of the network, are linked and each link may be characterized by values representing costs, profits, strengths or capacities. They model the transportation of commodities, which is subject to restrictions and costs given by the network structure. A commodity flow is represented by assigning a value, the transported amount, to each link. The most classical optimization problems ask for a maximum flow value of a single commodity subject to link capacities or for minimum cost flows.

Changing or developing the network structure may incur costs and be subject to further restrictions. If these changes affect the link properties only, e.g., a link's capacity or cost structure, then such models are referred to as capacitated network design or fixed-charge network flow. They are widely used for models not only in logistics but also in telecommunication and infrastructure planning. In the case where changes of the network structure concern the nodes of the network, or if the nodes have to be located, we are concerned with facility location or hub location models. These models capture the tradeoff between the costs for investing into the network structure and benefits for the resulting network flow such as cost savings or maximum throughput.

These classical mathematical models typically focus on optimization aspects that are considered central for real world problems. The real world is rarely simple or easy, yet simplifications are a necessity for purely mathematical models. Over the last decades, considerable progress has been made in the development of effective algorithms for many of theses models that, together with more powerful computing hardware, allows to successfully apply them to real-worlds problems of large scale. This trend of success stories naturally calls for more accurate models that are closer to reality. There are various possibilities along this path; e.g. we can extend classical models further by adding more details to the model, or we can integrate multiple optimization problems into one problem.

Many models typically assume that input data is precisely known and equal to some nominal values. However, in real-world applications there are uncertainties in the input data that may influence the quality or feasibility of the model. It is thus a natural quest to develop

models that are immune, as far as possible, to data uncertainty. Among the methodologies to deal with uncertainties, the concept of robust optimization has received considerable attention in the last decades and could successfully be applied to instances of larger scales.

This thesis explores possibilities and limitations of these ambitions in the context of logistics networks. We consider various network design, planning and flow problems and focus on theoretical insights as well as algorithms applicable in practice. The following aspects receive special attention:

Multidimensional Properties Classical multicommodity flow models consider one dimension of capacities or costs on the links, e.g., the mass of all commodities. We extend commodities by modeling their multidimensional properties, e.g. their mass and volume, to capture consolidation effects of commodities with complementing properties when considering transportation cost.

Integrating Temporal Decisions Classical network flow models are static, i.e. they output a static flow pattern that is oblivious of temporal constraints or effects. We integrate temporal decisions that affect the flow or the way we influence the network structure.

Cost Robustness for Demand Uncertainties One of the major concerns for uncertain data in logistics are demand values. A considerable effort is taken to obtain reliable forecasts, yet future development of the demands for transportation remains uncertain. Focusing on strategic models, we consider cost robustness, that is, uncertainties do not affect the feasibility of solutions but only its cost.

1.1 Outline and Contribution of the Thesis

In Chapters 2, 4 and 6 we introduce different optimization models for logistics networks. These chapters are self contained and also give an overview over relevant literature in the respective field. The models in Chapters 2, 4, and 5 emerged from within the research projects *MultiTrans* and *RobuNet* in close collaboration with logistics experts at 4flow AG [4fl17], a logistics consultancy company. We now give an overview over the chapters in this thesis. Parts of this thesis are already published in peer-reviewed journals, a publication remark is added in parenthesis for the respective chapter.

Chapter 2: Tactical Transportation Planning We propose a new mathematical model for transport optimization in logistics networks on the tactical level. Commodities are modeled with multidimensional properties. This allows to capture spatial consolidation effects as well as economies of scale with accurately modeled tariff structures. We also model temporal consolidation effects via a cyclic pattern expansion of the network. By using several graph-based gadgets, we are able to formulate the model as a capacitated network design problem. To solve the model, we propose a local search procedure that re-routes flow of multiple commodities at once.

Initial solutions are generated by various heuristics, relying on shortest path augmentations and LP techniques. As an important subproblem we identify the optimization of tariff selection on individual links, which we prove to be $\mathcal{NP}$-hard and for which we use exact as well as

fast greedy approaches. We complement our heuristics by lower bounds from an aggregated mixed integer programming formulation with strengthened inequalities. In a case study from the automotive, chemical, and retail industry, we prove that most of our solutions are within a single-digit percentage of the optimum. (This chapter is based on and in part identical to [Har+16]; parts of this chapter also appear in the PhD thesis of J. Matuschke.)

Chapter 3: Special Cases and Tractability Motivated by the model from Chapter 2, we consider multicommodity flow problems with multiple properties, where each commodity is required to be routed unsplittably. We focus on consolidation effects arising from the different properties of the commodities and relax economies of scale and capacity restrictions. Edge costs depend linearly on the maximum property usage of all commodities of the edge flow. We investigate problem variants and give hardness results for various special cases.

For the general case, we introduce the notion of forest costs that in general overestimate costs. Additionally we show that they are even exact for instances with only two consolidation layers. We also propose a dynamic program that finds solutions with optimal forest cost but has an exponential running time in the number of commodities.

Chapter 4: Robust Strategic Route Planning We devise a computational, robust optimization method for the strategic routing decisions of a logistics' customer, i.e., a company that uses the services of different freight forwarders to meet its transportation demands between several sources, sinks, and hubs. The costs of such transports are determined by tariff systems that typically show economies of scale and reward the consolidation of goods that complement each other in properties relevant for transport, such as weight and volume. In the strategic planning phase, routes and hubs have to be chosen roughly one year ahead, in particular, before the actual demand is known. Our method anticipates the fluctuation of demands by minimizing the worst-case cost over a restricted scenario set.

The combination of a realistic cost function, a robust modeling of uncertainty, and large-scale networks leads to highly intractable models. We show that the corresponding adversary problem is NP-hard. To nevertheless find solutions for real instances with very good worst-case cost we derive a carefully relaxed and simplified mixed integer linear program, that solves well for large instances due to its powerful linear programming relaxation. We test the method for real-world instances. The results show that robust optimization can significantly reduce worst-case cost. Furthermore, we derive from our method two heuristic techniques to solve even larger networks and report on the corresponding computational results.

Neglecting the typical uncertainty about demand values can cause significant cost in logistic routing problems. (This chapter is based on and in part identical to [RS16].)

Chapter 5: Hub Location for Logistics Networks This chapter extends the model from Chapter 4 to include hub decisions. We present an algorithmic toolkit to obtain robust solutions to the M-median hub location problem for logistics networks, thereby addressing two concepts for robustness: First, incremental hub chains are composed of solutions, one for each hub number, such that the hub sets of two adjacent solutions differ by at most one hub. They are thus robust under a changing number of hub nodes to operate. Second, cost robust solutions are robust for worst-case cost under uncertainty for demand values.

For incremental hub chains, we propose a flexible decremental framework that can be applied to large instances when combined with fast combinatorial heuristics. To assess the price of being incremental we compare incremental solutions with M-median solutions that are obtained with an LP-based hub search on two medium sized instances of our test set.

Identifying the large number of demands as algorithmic bottleneck for large instance, we present a generic aggregation concept and test two aggregation techniques. This way we obtain incremental hub chains even for the largest instance. To the best of our knowledge, this has not been achieved for instances of this size before.

The LP-based hub search relies on an approximation of the cost function, which allows to solve the cost robust model for medium sized instances. We evaluate this method for various uncertainty sets derived from historical data. We also provide an a posteriori evaluation with worst case cost for the exact cost function to observe the price of robustness. For our instances, we find a price of robustness that is moderate for interval uncertainty sets that use average demand values as lower bounds. Roughly one percent loss for historical average costs is compensated by two or four percent gain of robust solutions for their specific worst case costs. For trend based uncertainty sets however, which are obtained with an outlier methods, this tradeoff increases: Here, three percent loss in historical average cost can save up to ten percent for worst case costs.

Chapter 6: Scheduling Maintenance Jobs in Networks In this chapter, we investigate the problem of scheduling the maintenance of edges in a network, motivated by the goal of minimizing outages in transportation or telecommunication networks. We focus on maintaining connectivity between two nodes over time. For the special case of path networks, this is related to the problem of minimizing the busy time of machines.

We show that the problem can be solved in polynomial time in arbitrary networks if preemption is allowed. If preemption is restricted to integral time points, the problem is NP-hard and in the non-preemptive case we give strong non-approximability results. Furthermore, we give tight bounds on the power of preemption, that is, the maximum ratio of the values of non-preemptive and preemptive optimal solutions.

Interestingly, the preemptive and the non-preemptive problem can be solved efficiently on paths, whereas we show that mixing both leads to a weakly NP-hard problem. Additionally, we can give a simple 2-approximation for the latter problem. (This chapter is based on and in part identical to [Abe+17]; parts of this chapter also appear in the PhD thesis of R. Rischke.)

Chapter 7: Conclusion In this chapter we conclude the results of this thesis.

Notations We assume the reader to be familiar with the basic topics of combinatorial optimization and refer to the textbooks [KV12; Sch03]. Wherever possible, we try to keep the notations in this thesis conform to those used in those books. Further notation is introduced where necessary; we also point to the list of notations in the appendix, see page 225.

Chapter 2

Tactical Transportation Planning

2.1 Introduction

The task of designing and operating logistics networks belongs to the broad realm of *supply chain management (SCM)*, "the management of flows between and among all stages of a supply chain to maximize total profitability" [CM07]. As this very general definition indicates, SCM addresses a multitude of issues ranging from location, product, and marketing decisions to the management of information exchange and coordination across different stages of the supply chain. *Transportation planning* in particular occupies a central place in SCM, as transport and storage of physical goods account for a significant share of the operational cost in a supply network.

Due to a strong variance in lead times associated with the different decisions to be made in SCM, the planning process is naturally structured hierarchically in strategic, tactical, and operational levels [SKS03]. The work presented in this chapter is concerned with *transportation planning on the tactical level*. Here, it is commonly assumed that the supply chain is already in place: location and product decisions have been made, and the general design of the supply chain network is fixed. Typical logistic decisions on the tactical level include the amount of flow between the existing nodes of the network, e.g., which customers to serve from which warehouses or suppliers, how much inventory to keep at which locations, and which transportation modes and delivery frequencies to employ on the different connections [GP03].

We proposes an approach to model and solve the key tasks in tactical transportation planning in an integrated fashion, explicitly including realistic transport tariffs and the trade-off between inventory cost and economies of scale in transportation.

2.1.1 Problem Description

We proceed to give a general description of the task we refer to as tactical transportation planning and introduce some terminology we will use throughout this chapter. We consider a network of *facilities*, which are of different types, like production plants, warehouses, distribution centers, or retailers. Some facilities have a *supply* of, or a *demand* for certain products, also known as *commodities*, which can be numerous and very different, e.g. in their mass, volume, or value. Facilities are joined by *transport relations*, and on each transport relation, different *transport tariffs* are available corresponding to concurring offers of freight

forwarders and available transportation modes. Each transport tariff is characterized by capacity restrictions and a cost function, describing how much of a commodity (or of some commodity mix) can be transported, and the cost incurred for a given amount of a commodity (mix). E.g., a full truck load tariff may have a certain truck type's payload and footprint as capacity restrictions and incur a fixed charge cost. Some facilities may be able to carry *inventory*, usually with a commodity-dependent capacity and cost. *Handling cost* may result from commodities passing through a facility, like a distribution center, regardless of whether they are moved to inventory or not.

Quite commonly, transportation cost includes fixed-charge costs for dispatching shipments, and the larger a shipment, the lower the effective per-unit shipping cost. Hence, a key ingredient to successful tactical planning in a logistics network is the efficient *consolidation* of material flows, i.e., the combination of smaller order amounts into larger shipments in order to utilize capacity efficiently and enable economies of scale [Çet05]. Consolidation may occur over space as well as over time. In *spatial* consolidation, material flows of different origins are accumulated at one node and forwarded jointly to the next. In *temporal* consolidation, material is kept in inventory at a node for some time in order for more flow to arrive, thereby enabling a larger outbound shipment. Since holding inventory also incurs cost, however, there is a tradeoff to be considered here.

This interplay between inventory cost and different transport tariffs necessitates a notion of time in planning. Since temporal details such as transport transit times or demand deadlines are commonly postponed to operational planning, the goal in tactical optimization is a cyclic *pattern* of deliveries and inventory. The length and structure of this pattern usually follows some natural notion of rough timing, like "once every month", "once every week" or "once every day of the week", and in each slot of the pattern (like in one month, week or weekday), deliveries are dispatched, and inventories are replenished or depleted.

All in all, the outcome of tactical transportation planning as described here comprises

- the paths each commodity takes through the network from its sources to its sinks, i.e., the total amount of flow for each commodity on each transport relation,
- the transport tariffs employed on each transport relation, together with an assignment of a commodity mix to each of them,
- a cyclic pattern in which transports are executed for each tariff used on each transport relation, including the amounts shipped for each commodity in each slot of the pattern, and finally
- a pattern of inventory levels for each commodity at each node, supporting the above transport patterns.

Again, note that in tactical planning, the aim is not to use the results to operate the logistics network directly, as this is the subject of operational planning. Rather, tactical optimization intends to aid with decisions which have to be made with some lead time, providing the framework for efficient operation: How much throughput capacity needs to be reserved at certain distribution centers? Which logistics provider should be cooperated with on which network connections, and which available tariffs will be employed on what volume of commodities? Hence, the main purpose of many details in tactical modeling is not primarily to reflect operational reality, but much more to yield a realistic assessment of operational cost in the framework provided.

2.1.2 Our Contribution and Overview of the Paper

In Section 2.2, we propose a new model for the optimization of transportation networks on the tactical level. In our model, different commodities are flexibly characterized in terms of their properties (like mass, volume, and value), and a choice of many different transportation modes and tariffs is naturally incorporated, with capacities and costs accurately reflecting the properties of the (mix of) commodities transported. Moreover, our model includes the possibility for flexible, cyclic delivery patterns on each network connection, accurately modeling the tradeoff between inventory cost and economies of scale in transportation. While we assume that location decisions have been made and facilities are already in place, the main decision variables of our model include the flow paths of commodities through the network, the transportation tariffs, and inventory levels. By using several graph-based gadgets, we are able to formulate our problem as a network design problem. Note that in contrast to the broad literature on classical network design problems (see the next Section 2.1.3 for concrete pointers to the literature) our formulation integrates different realistic transportation tariffs, cyclic delivery patterns, and inventory costs all in one model.

While the resulting network design problem can be naturally formulated as a mixed integer programming (MIP) model, the precise replication of complex tariff structures (via the previously mentioned gadgets) leads to a drastically increased number of variables putting basic MIP approaches out of reach (at least for instances arising in practice). We identify the problem of selecting optimal tariffs on a single transport relation as an important subproblem that is crucial in speeding up the solution process: In order to identify cost efficient paths, our algorithms need good and fast estimates on the cost incurred by sending a particular amount of flow along a transport relation. These cost estimates are performed very frequently (easily more than a million times during the optimization of a single network) and therefore need to be carried out even faster. In Section 2.3, we will propose different algorithms that provide an efficient balance of accuracy and speed for solving this $\mathcal{NP}$-hard subproblem.

In Section 2.4, we then propose a local search heuristic that employs local changes on a path decomposition of flow in the network using the previously mentioned tariff selection subroutines. In contrast to many local search heuristics known in the literature (that either work directly on the design variables or reroute flow of a single commodity only), our approach applies a neighborhood search based on path decomposition of flow and re-routing multiple commodities simultaneously. In order to obtain good initial solutions for our local search heuristic, we provide two successive shortest path type algorithms. The first method is designed with an emphasis on speed and low memory requirement, being able to generate solutions of reasonable quality for even the largest instances in short time. The second is more accurate in cost estimation and is therefore used as the central subroutine in our local search improving moves. By forbidding certain paths (for instance direct connections) and linearizing costs we further tune the initial solutions towards a high level of flow consolidation that will eventually be disaggregated by the local search heuristic.

In Section 2.5, we complement our heuristic approach by mixed integer programming techniques. As the plain MIP formulation is not suited for solving reasonably sized real-world instances due to enormous problem sizes, we propose an aggregated formulation that considerably reduces model size and still yields good dual bounds. We combine this with efficient preprocessing techniques to tighten the relaxation and a post-processing step to improve solution quality. Combining the LP relaxation of this strengthened and aggregated formulation

with the tariff selection heuristics mentioned earlier yields a third way of constructing initial solutions for our local search procedure, which shows best final results on average.

In Section 4.8, we evaluate the performance of our different algorithmic approaches on a library of real-world instances provided by our project partner 4flow AG [4fl17], a logistics consultancy company. The test set consists of case studies from the automotive, chemical, and retail industry with up to thousands of locations and hundreds of commodities. We can prove that most of our solutions are within a single-digit percentage of the optimum, and that our modelling and algorithmic techniques yield a cost reduction of over ten percent over the current status quo, which could result in annual savings of several millions of euros.

2.1.3 Related Work

Mathematical optimization for logistic problems has been a vast field of research for several decades. We give an overview over models and algorithms for tactical transportation planning.

Models for Transportation Planning

An excellent overview of network-based optimization techniques for SCM is given by [GP03]. The authors review articles dealing with strategic as well as tactical and operational planning.

In one of the earliest optimization models for SCM by Geoffrion and Graves [GG74], the authors model a multicommodity network with several plants, possible distributions center locations, and demand zones on the strategic level. While the model incorporates fixed location costs, as well as upper and lower bounds on the throughput of a distribution center, it does not consider inventory decisions and assumes transportation costs to be linear. The resulting mixed integer programming (MIP) model is solved using Benders decomposition. A strategic optimization model that incorporates the interdependence of location, transportation, and inventory decisions is described by [Jay98]. Here, different transportation modes can be chosen for each connection in the network. Each mode is associated with a commodity-dependent per-unit cost and a *delivery frequency*. Keeping inventory at a plant or warehouse incurs per-unit inventory cost, and the amount of inventory held results from the delivery frequencies of the outbound transportation modes used. Note that this still captures temporal consolidation rather coarsely, as theoretically, also transportation modes with low delivery frequency could carry low shipping volume, making their assumed low per-unit cost unrealistic. The model is solved using standard MIP solvers.

While the above network-wide SCM models are focused on strategic planning and incorporate location decisions, the tactical and operational tradeoff between transportation and inventory cost lies at the heart of *dynamic lot-sizing* in inventory theory. In the basic version of dynamic lot-sizing introduced by [WW58], different demands for a commodity at one facility need to be met in multiple periods. In each period, an arbitrary amount can be ordered at fixed per-order cost, while per-unit inventory cost is incurred. The goal is to determine the amount ordered in each period such that all demands are met on time and the sum of order and inventory cost is minimized. This basic model has been extended in many ways since then, and most variants are computationally hard, see e.g., [JD07] for an overview. The practical importance of considering the trade-off between transportation and inventory cost is highlighted impressively by [Bur+85] and [Blu+87], where the authors were able to reduce logistics cost by 26% in a case study for General Motors.

Generalizing lot-sizing to networks with multiple stages brings it closer to the requirements of tactical transportation planning. The first such model was introduced by [CS60] and further developed by [AGK84; AG86]. An overview of more recent works can be found by [Sta03]. Most of these models, however, still make rather restrictive assumptions on the structure of the network considered and transportation costs incurred. Moreover, the quantity of material flowing between node pairs is fixed a priori in all lot-sizing models, so the possibility for more spatial consolidation at hubs is effectively ignored.

[KKS10] propose a general model for the integrated operational planning of external and internal logistics of the last two stages of a supply chain. In their model, all costs depend on the usage of resources, like mass or volume, and this dependence can be piecewise constant as well as linear and may involve multiple resources. Planning occurs over multiple however non-cyclic periods, and in particular, inventory cost is taken into account. The authors devise a flow-based construction heuristic to generate an initial feasible solution that is passed to a standard MIP solver. In order to introduce all details necessary for realistic operational planning, their model even allows for logical relations between different resources, which however significantly increases the algorithmical challenge of solving large scale instances. Accordingly, their solution approaches are validated on relatively small instances involving only five planning periods with networks of up to 25 nodes, several hundred edges, and up to one hundred commodities.

In a more tactical context, [SKS10] propose a similar resource-based model for optimizing the choice of delivery profiles in *area forwarding based networks*. In such networks, suppliers are grouped into areas and each area is equipped with a consolidation center run by a logistics carrier. The main decision variables are the choices from a fixed set of delivery profiles for each supplier and the usage of vehicles on the main legs (i.e., the connections between consolidation centers and the target). The authors propose a solution method that first decomposes the model by fixing certain decisions for each possible delivery profile and then generates an initial feasible solution for the MIP solver using a two-phase construction heuristic. The approach is evaluated in the logistics network of a German truck manufacturer, achieving cost savings of up to 36% in individual areas.

The model introduced in this chapter, as well as the models by [KKS10] and [SKS10] are based on capacitated network design formulations (see below). An alternative approach to modelling non-linear transportation tariffs are concave-cost network flows, see [GP90] for a survey. Note that also all three models mentioned above include the possibility of concave cost functions (cf. Section 2.2.4 to see how they can be modelled in context of the present work).

In contrast to the the model of [KKS10], our approach focuses exclusively on transportation planning. It thus does not consider globally interdependend resources, making it possible to encapsulate tariff selection in a local subproblem and allowing for transportation networks of larger size to be solved. It also differs from the model by [SKS10], which employs delivery profiles to model replenishment cycles, while the present work is concerned with *dynamic planning* and also allows for more general networks with multiple levels of intermediate hubs instead of two-layered area forwarding based networks.

Capacitated Network Design

While network flow seems to be the dominant aspect in logistics network optimization, the fixed cost nature of transportation brings in network design decisions: We have to install sufficient capacity in the network such that all flow can be routed. In literature, such mixtures of network flow and network design are referred to as *capacitated network design* or *fixed-charge network flow*, and are widely used for models not only in logistics but also in telecommunication and infrastructure planning (see the surveys by Magnanti and Wong [MW84] and Crainic [Cra00]).

Most capacitated network design problems are very hard to solve both in theory and practice. In fact, the model presented in chapter generalizes several problems that are not only $\mathcal{NP}$-hard but even highly inapproximable from a theoretical point of view, e.g., the single-pair version of the capacitated survivable network design problem, for which [Cha+11] showed that it does not even permit an approximation factor of $2^{\log^{1-\varepsilon}(n)}$ for any $\varepsilon > 0$ (unless all problems in $\mathcal{NP}$ can be solved in quasipolynomial time). Furthermore, $\mathcal{NP}$-hardness still holds for very basic and sparse classes of networks like so-called series-parallel graphs because a version of the multiple Steiner subgraph problem [RP86] can be reduced to our model.

This intrinsic hardness, combined with the enormous size of instances encountered in practical applications from logistic contexts, leaves little hope for exact solution approaches that run in acceptable time. Therefore, fast combinatorial heuristics appear to be the method of choice. The current state of the art is mainly built on specialized tabu search procedures. Crainic, Gendreau, and Farvolden [CGF00] proposed a tabu search procedure based on a neighborhood in the multicommodity flow polytope. Their algorithm has later been adapted for parallelization by Crainic and Gendreau [CG02]. A different neighborhood for tabu search was introduced by [GCG03], operating on the network design and modifying it along cycles. This procedure has been refined by the same authors by supplementing it with a path relinking technique [GCG04].

A different approach for solving fixed-charge network flow problems is constituted by *slope scaling*. The slope scaling procedure, first proposed by Kim and Pardalos [KP99] for single-commodity fixed-charge network flow, iteratively solves the min-cost flow problem arising from linearizing the fixed costs according to the current solution. Crainic, Gendron, and Hernu [CGH04] generalize this technique to multicommodity capacitated network design, and augment it by Lagrangian perturbation and intensification/diversification mechanisms based on a long-term memory.

All algorithms referenced above are designed for general capacitated network design problems and have been successfully tested on a standard benchmark set of randomly generated instances of moderate size with at most 100 nodes and 400 edges, introduced by Crainic, Gendreau, and Farvolden [CGF00].

MIP Approaches to Network Design

Several exact solution techniques for capacitated network design have been studied, see e.g., the survey by Costa [Cos05]. These techniques range from Lagrangean relaxation over column generation to Benders decomposition. Kliewer and Timajev [KT05] integrate cover inequalities and local cuts in a Lagrangean-based lower bound, whereas Frangioni and Gendron [FG09]

study a 0-1-reformulation for piecewise linear costs and show the computational benefits of strong linking inequalities. Chouman, Crainic, and Gendron [CCG11] present lifting procedures for *strong capacity* and *network cutset inequalities* for fixed-charge network flow problems. Another promising technique to solve capacitated network design problems is to apply a Benders decomposition, see for instance [Cos05; Cak09]. [CCG09] show the relation between different classes of inequalities. In particular, the authors explain how the inequalities from (non-extreme) dual rays of the Benders framework and cutset inequalities can be strengthened via shortest path computations to become metric inequalities. To improve the running times, Fischetti, Salvagnin, and Zanette [FSZ10] propose to find a minimal infeasible subsystem. They show that this idea can be integrated into the subproblem heuristically.

These works indicate that the scope of tractable instance sizes for these methods is roughly limited to 30 nodes, 500 edges and 200 commodities, i.e., for the few larger instances reported on, the provable gaps on solution quality exceed single digits.

2.2 Mathematical Model

Our model, which we call *TTP* for tactical transportation planning, is at its heart based on multicommodity network flow, with both linear and fixed-charge cost on the edges. However, we extend the standard concepts of capacity and cost to more generality in order to reflect the requirements of logistics modelling more precisely. Moreover, we expand the underlying network significantly in order to model delivery patterns, inventory effects, and complex transport tariffs. We proceed to detail all of these features in the following sections.

2.2.1 Pattern Expansion

The tradeoff between minimizing inventory cost and taking advantage of the economies of scale in transportation is of key importance in tactical logistics planning. Temporal and spatial consolidation effects regularly determine which tariff is most suitable on a connection. Consequently, even the decision which path in the network is most efficient for a commodity may ultimately depend on temporal delivery patterns. As tactical planning defines the environment for operational planning which will take place again and again over time, a solution should be a *cyclic* pattern for dispatching deliveries and replenishing and depleting inventories. To integrate temporal and spatial consolidation together with cyclic delivery patterns, we introduce the notion of *pattern expanded networks*.

A pattern expanded network denoted by $\mathcal{G}$ has two main components: The first is the *base network* B, which comprises the physical entities of the transport network: facilities (or *nodes*) together with corresponding *transport relations* between facilities. The second parameter is a cycle length F defining the number of time slots (e.g., 7, 30, or 356 days) available in a period. The pattern expanded network $\mathcal{G}$ is now obtained from B and F by introducing F copies of B denoted by $B_1, \ldots, B_F$ and connecting copies of each node of every two adjacent networks B_i and B_{i+1} by directed *holdover edges* (the direction is from nodes in B_i to those in B_{i+1}). Moreover, the nodes of the last copy B_F are also connected by holdover edges to their corresponding copies in the first copy B_1, thus, giving a cyclic network structure. If commodities are sent along holdover edges from B_F to B_1, this corresponds to storing commodities at the corresponding nodes at the end of a cycle, to the beginning of the next cycle. Costs can be associated with holdover edges modeling inventory costs. In the following

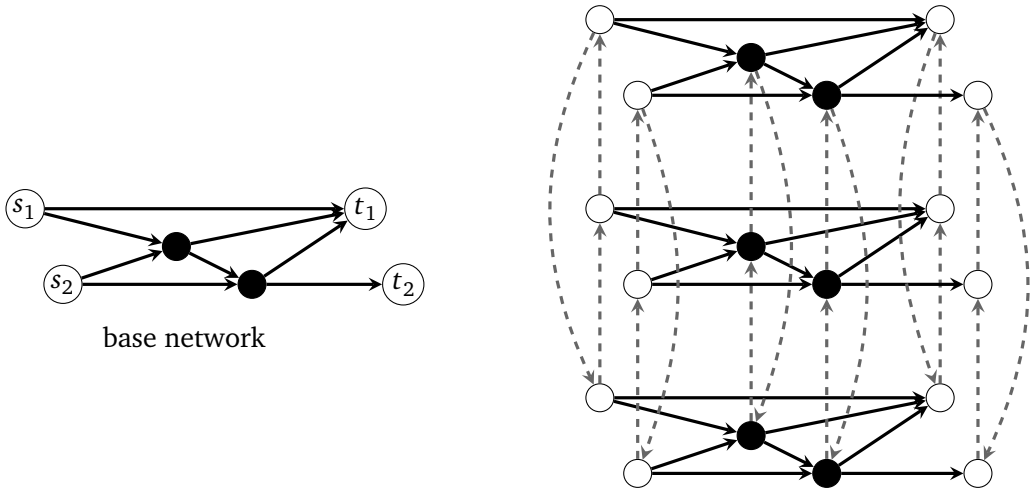


Figure 2.1: Network Expansion. Base network with associated pattern expanded network. Dashed edges denote holdover edges.

we will conceptually not differentiate between holdover edges and transport edges. We denote the set of nodes in the pattern expanded network by $\mathcal{V}$ and the set of all edges (also called transport relations) of $\mathcal{G}$ by $\mathcal{R}$.

We illustrate this cyclic construction with an example. Consider the base network in Fig. 2.1(a) involving two source-sink pairs (s_1, t_1) and (s_2, t_2). In this example, we chose $F = 3$, i.e., transports may occur only in three time slots, e.g., three days a week. The pattern expanded network now involves the three copies of the base network and the additional holdover edges as illustrated in Fig. 2.1 (b).

2.2.2 Commodities and Properties

Commodities in a logistics network can be very diverse, e.g., in their size, weight, or value, and logistic costs and transport capacities cannot be realistically assumed to be oblivious to this diversity and the resulting interdependencies when mixing commodities in transport. We introduce the concept of flexible *properties* to characterize commodities. A set of commodities K and a list of relevant properties P are parameters of our model. Each commodity $i \in K$ is assigned a per unit extent α_{ij} for each property $j \in P$. The main motivation for introducing these properties is that transportation costs introduced in the next section will mostly depend on the total extent of each property of a commodity mix (rather than the specific type of commodities itself), thus reflecting the effects of consolidating goods for utilizing vehicle capacities more efficiently.

In the following, a mix of commodities will be denoted by a *commodity vector* $x \in \mathbb{R}_+^K$ and the *aggregated properties* of such a mix x is expressed by $\alpha(x) \in \mathbb{R}_+^P$ with $\alpha_j(x) := \sum_{i \in K} \alpha_{ij} x_i$.

Each node in the pattern expanded network may have a supply of, or a demand for certain commodities. These supplies and demands are expressed by a balance vector $b(v) \in \mathbb{R}^K$ for each node $v \in \mathcal{V}$ (note that these values might be different even for distinct copies of the same node in the base network). A node with a supply ($b_i(v) > 0$) of a certain commodity $i \in K$ is called a *source* of i, a node with a demand ($b_i(v) < 0$) is called a *sink* of i. The goal

is to transport all supplies from the sources to the sinks, satisfying all demands.

2.2.3 Transport Tariffs

When shipping goods on a transport relation, different *transport tariffs* are available. For each transport relation $R \in \mathcal{R}$ we denote by $T(R)$ the set of available tariffs for transporting a flow of commodities from start(R) to end(R). Each such tariff $t \in T(R)$ is associated with a cost function $C_t : \mathbb{R}_+^K \to \mathbb{R}_+$. We also assume that all cost functions fulfill the economies of scale principle, i.e.,

$$C_t(a+b) \leq C_t(a) + C_t(b). \tag{2.1}$$

A solution of our model consists of a multicommodity flow in the pattern expanded network satisfying all demands, together with an assignment of the flow on each transport relation to the tariffs available on this relation. More formally, let $x(R) \in \mathbb{R}_+^K$ denote the total (multicommodity) flow to be shipped on transport relation $R \in \mathcal{R}$, and let $x(t) \in \mathbb{R}_+^K$ denote the amount of flow transported using tariff $t \in T(R)$. Then our goal is to find an optimal solution to

$$\begin{aligned}
\min \quad & \sum_{R \in \mathcal{R}} \sum_{t \in T(R)} C_t(x(t)) \\
\text{s.t.} \quad & \sum_{R \in \delta^+(v)} x_i(R) - \sum_{R \in \delta^-(v)} x_i(R) = b_i(v) && \forall\, v \in \mathcal{V},\ \forall\, i \in K \\
& \sum_{t \in T(R)} x_i(t) = x_i(R) && \forall\, R \in \mathcal{R},\ \forall\, i \in K \\
& x(t) \geq 0 && \forall\, t \in T(R),\ \forall\, R \in \mathcal{R}
\end{aligned}$$

where $\delta^+(v)$ and $\delta^-(v)$ denote the sets of outgoing and incoming arcs of node v, respectively.

We will now present a set of cost functions that covers most tariffs occurring in today's logistical applications. In the next section, we will then show how all these cost functions can also be modelled in a unified form as a capacitated network design problem.

Linear costs. In many logistical applications, commodity-dependent linear costs of the form

$$C(x) = \sum_{i \in K} c_i \cdot x_i$$

with cost rates $c_i \in \mathbb{R}_+$ for each commodity occur, e.g., in the form of handling costs, in-stock and in-transit inventory costs and simple linear tariffs without interdependencies of the transported commodities.

Maximum over multiple cost rates. Tariffs can also be specified as the maximum over varying cost rates for distinct properties, i.e., when sending a shipment that rate applies for which the cost is highest. More formally, with c_j being the cost rate for property j, the cost function is given as

$$C(x) = \max_{j \in P} c_j \cdot \sum_{i \in K} \alpha_{ij} x_i.$$

Note that, in contrast to the linear costs described in the preceding paragraph, these maximum cost functions capture the effect of cost savings when mixing commodities of different dimensions, e.g., light but voluminous with heavy but compact ones.

Property-dependent piecewise constant costs. Many tariffs, such as those offered by most full truck load (FTL) carriers and some less than truck load (LTL) carriers, are based on piecewise constant cost functions, i.e., they are specified by a cost $c \in \mathbb{R}_+$ and a capacity vector $\beta \in \mathbb{R}_+^P$ for a single shipment, yielding the function

$$C(x) = c \cdot \max_{j \in P} \left\lceil \frac{\alpha_j(x)}{\beta_j} \right\rceil.$$

In practice, logistic carriers offer groups of such tariffs realizing different levels of discount for higher shipment volumes. We will see in Section 2.3 that finding the most cost-efficient combination of such tariffs for a given shipment volume is already an NP-hard problem.

Of course, linear and fixed costs can also occur at the same time, e.g., to model a transport to a distribution center which incurs fixed cost for transportation and a linear cost for handling the incoming shipment at the distribution center. We thus also allow the combination of these two cost types.

Incremental discount costs. We consider a tariff with varying cost rates depending on a single property. The cost rates are specified on intervals and decrease with increasing size of shipment, resulting in a piecewise linear and concave cost function; see Table 2.1 for an illustration. Formally, label the intervals from 0 to L. For each $\ell \in [L]$, let $c^{(\ell)} \in \mathbb{R}_+$ be the cost rate on the interval $\left[\beta_j^{(\ell)}, \beta_j^{(\ell+1)}\right)$ for the fixed property $j \in P$, with $0 = \beta_j^{(0)} < \beta_j^{(1)} < \ldots < \beta_j^{(L)} < \beta_j^{(L+1)} = \infty$ and $c^{(0)} > c^{(1)} > \ldots > c^{(L-1)} > c^{(L)}$. Then the cost function is

$$C(x) = \sum_{\ell=0}^{L} c^{(\ell)} \cdot \min\left\{\beta_j^{(\ell+1)} - \beta_j^{(\ell)},\ \left(\alpha_j(x) - \beta_j^{(\ell)}\right)^+\right\}.$$

All-unit discount costs. Again we consider linear cost rates in some property $j \in P$ with several levels of decreasing per-unit cost rates. Different from the above, however, a cost rate applies to the entire transport volume as long as it lies within the corresponding interval. To ensure monotonicity, a cost cap applies whenever the cost with respect to the current rate exceeds the cost at the beginning of the next level—this corresponds to the common practice of declaring higher volumes than actually transported in such cases [Cha+02]. See Table 2.1 for a graphical illustration of the resulting cost function. Formally, if cost rate $c^{(\ell)}$ for $\ell \in [L]$ is applicable starting from transport volume $\beta_j^{(\ell)}$ on, the cost function is

$$C(x) = \min_{\ell \in [L]} \left(c^{(\ell)} \cdot \max\left\{\alpha_j(\tilde{x}),\ \beta_j^{(\ell)}\right\}\right).$$

2.2.4 Reformulation as Capacitated Network Design

We will now provide a different perspective to the model presented in the previous section. We introduce the concept of *containers* to model the different types of tariffs in a way that leads to a unifying description of the above model as a fixed-charge multicommodity flow problem. A natural formulation as a mixed integer program can easily be obtained from this description, making it accessible to MIP based solving techniques, while its compact structure effectively demonstrates the degree of mathematical uniformity achieved in modelling.

We will first present the alternative formulation of the model to its full extent, and then show the equivalence to the formulation in the previous section by describing how different cost functions can be modelled using containers.

The Tariff Expanded Network

For each tariff on a transport relation, we introduce a *gadget* consisting of edges, which connects the start node of the relation with its end node. On each edge, a certain type of *container* is available, and capacities can be installed on the edge in increments of this container type. After replacing all transport relations in the pattern expanded network by the corresponding gadgets for their tariffs, we obtain the *tariff expanded network* $G = (V, E)$ consisting of the nodes of the pattern expanded network, the additional nodes introduced in the gadgets and the edges introduced in the gadgets. Each container of edge e has a capacity for every property. A solution to the container-based formulation of our model specifies for each edge e the (integer) number of containers $y(e)$ installed at e together with the edge flow values $x_i(e)$ for each commodity i. For each property, the capacity installed at e must be sufficient to transport the flow. More formally, recall that α_{ij} denotes the per-unit extent of commodity i w.r.t. property j, and let $\beta_j(e)$ be the corresponding capacity of a container at edge e. Then the *capacity constraints*

$$\sum_{i \in K} \alpha_{ij} x_i(e) \leq \beta_j(e) y(e) \quad \forall\, j \in P \tag{2.2}$$

must hold at every edge $e \in E$. Moreover, an upper bound $u(e)$ on the number of containers installed on an edge e may be specified.

In a feasible solution, the multicommodity flow x has to satisfy all demands. We extend the node balances introduced for the nodes in the pattern expanded network by setting the balances for all nodes artificially introduced by tariff expansion to zero for each commodity. We thus obtain the *flow conservation constraints*

$$\sum_{e \in \delta^+(v)} x_i(e) - \sum_{e \in \delta^-(v)} x_i(e) = b_i(v) \quad \forall\, i \in K \tag{2.3}$$

that must be valid at every node $v \in V$ of the tariff expanded network.

For each container installed at e, a fixed cost $c(e)$ has to be paid. Flow sent along e may furthermore incur a commodity dependent linear cost $c_i(e)$ (which may naturally be used to model property dependent linear costs as well). Thus, the total cost of a solution is

$$\sum_{e \in E} \left(c(e) y(e) + \sum_{i \in K} c_i(e) x_i(e) \right).$$

Putting all of this together, the fixed-charge multicommodity flow problem resulting from the container formulation can be directly formulated as a MIP.

$$
\begin{aligned}
\min \quad & \sum_{i\in K}\sum_{e\in E} c_i(e)x_i(e) + \sum_{e\in E} c(e)y(e) & \\
\text{s.t.} \quad & \sum_{e\in\delta^+(v)} x_i(e) - \sum_{e\in\delta^-(v)} x_i(e) \;=\; b_i(v) & \forall v\in V, i\in K \\
& \sum_{i\in K} \alpha_{ij}x_i(e) \;\le\; \beta_j(e)y(e) & \forall e\in E, j\in P \\
& y(e) \;\le\; u(e) & \forall e\in E \\
& x_i(e)\in\mathbb{R}_+,\; y(e)\in\mathbb{Z}_+ & \forall e\in E, i\in K
\end{aligned}
$$

Note that a flow in the tariff expanded network (i.e., on edges) can be transformed into a flow in the pattern expanded network (i.e., on transport relations) by setting $x(t)$ to be the amount of flow going from start(R) to end(R) through the gadget corresponding to t, which corresponds to the total amount shipped using this tariff.

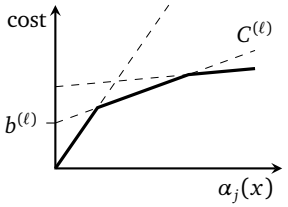
The gadget of each tariff t will be designed to model its cost function C_t in the sense that the cost incurred by the flow in the gadget (in terms of required container capacity and linear costs) equals $C_t(x(t))$. Therefore, the total cost of the solution in the tariff expanded network equals the cost of the flow in the pattern expanded network.

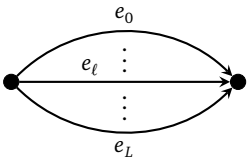
Modelling Tariffs with Containers

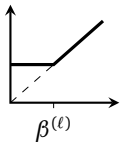
We now proceed to explain how containers can be used to accurately model the different types of transportation tariffs introduced in the previous section; see Table 2.1 for an overview of the more complex gadgets.

Modeling linear and piecewise constant costs. It is clear that both commodity-dependent linear costs and property-dependent piecewise constant costs are directly captured by the container concept. Linear costs are part of the definition, while piecewise constant tariff groups can be directly modeled by introducing a bundle of parallel edges, one for each tariff in the group. The container on each edge takes the capacity and cost of the corresponding tariff.

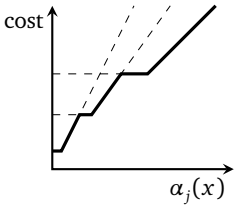
Modeling the maximum over multiple cost rates. In order to model the maximum over multiple cost rates we need to introduce *fractional containers* to the model, i.e., the variable $y(e)$ corresponding to the number of installed copies of such a container can be fractional. We use a single gadget edge for each tariff that corresponds to a maximum over multiple cost rates c_j with $j\in P$. We set the cost to $c(e)=1$ and the capacity $\beta_j(e)=1/c_j$ for each $j\in P$. Sending a flow of $x(e)$ through this edge requires $y(e)$ to be set to $\max_{j\in P}\alpha_j(x(e))/\beta_j(e)$, which is equal to the cost function by choice of $\beta_j(e)$. Note that introducing such fractional containers does not have significant impact on the complexity of the model. Still, for the sake of simplicity, we will assume throughout this work that all containers have to be installed in integral increments.

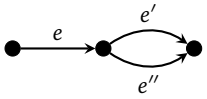
<table>
<tr><th>tariff</th><th>cost</th><th>gadget</th></tr>
<tr><td>incremental discount
(piecewise linear concave)</td><td> 

$C(x) = \min_{\ell \in [L]} C^{(\ell)}(x)$
$C^{(\ell)}(x) := c^{(\ell)}\alpha_j(x) + b^{(\ell)}$</td><td>minimum modeled
by parallel edges
 

$c(e_\ell) = b^{(\ell)}$
$c_i(e_\ell) = \alpha_{ij}c^{(\ell)}$</td></tr>
<tr><td>all-unit discount</td><td> 

$C^{(\ell)}(x) := c^{(\ell)} \cdot \max\{\alpha_j(x),\ \beta^{(\ell)}\}$

$C(x) = \min_{\ell \in [L]} C^{(\ell)}(x)$</td><td> 

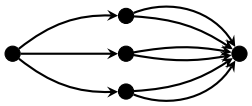
<table>
<tr><th></th><th>e</th><th>e'</th><th>e''</th></tr>
<tr><td>c</td><td>$c^{(\ell)}\beta^{(\ell)}$</td><td>0</td><td>0</td></tr>
<tr><td>c_i</td><td>0</td><td>$\alpha_{ij}c^{(\ell)}$</td><td>0</td></tr>
<tr><td>β_j</td><td>∞</td><td>∞</td><td>$\beta^{(\ell)}$</td></tr>
</table>
minimum modeled
by parallel gadgets
  </td></tr>
</table>

Table 2.1: Modelling complex transport tariffs with containers

Modeling incremental discounts. Piecewise linear concave functions arising from incremental discount tariffs can be interpreted as the minimum of several affine linear functions. Again denoting the linear segments of the function by 0 to L with cost rates $c^{(\ell)}$ and break points $\beta^{(\ell)}$, we define

$$C^{(\ell)}(x) := c^{(\ell)}\alpha_j(x) + b^{(\ell)} \quad \text{with} \quad b^{(\ell)} := \sum_{k=0}^{\ell-1}(c^{(k)} - c^{(\ell)})(\beta_j^{(k+1)} - \beta_j^{(k)})$$

for $\ell \in [L]$. It is easy to verify that $C_t(x) = \min_{\ell \in [L]} C^{(\ell)}(x)$; see Table 2.1 for an illustration. We now introduce a gadget of $L+1$ parallel edges $e_0, \ldots, e_L$ with $c(e_\ell) = b^{(\ell)}$ and $c_i(e_\ell) = \alpha_{ij}c^{(\ell)}$. Sending flow along edge e_ℓ incurs the cost $C^{(\ell)}$ and an optimal solution will always send flow along that edge which achieves the minimum cost for the transported amount.

Modeling all-unit discounts. Note that functions of the form $c^{(\ell)} \cdot \max\{\alpha_j(x),\ \beta^{(\ell)}\}$ can be modeled by the following gadget; also see the corresponding figure in Table 2.1. Introduce

a series-parallel graph, consisting of a single edge e followed in series by two parallel edges e' and e''. We set the fixed costs $c(e) = c^{(\ell)}\beta^{(\ell)}$ and $c(e') = c(e'') = 0$. We also set the linear costs $c_i(e) = c_i(e'') = 0$ and $c_i(e') = \alpha_{ij}c^{(\ell)}$ for all $i \in K$. Capacity $\beta_j(e'')$ is set to $\beta^{(\ell)}$, all other capacities are left infinite, and we let $u(e'') = 1$ so that only one container can be installed on e'', while the number of containers remains unbounded for all other edges. Now, all-unit discount tariffs, which can be represented as minimum of such functions, can be modeled by introducing several of these gadgets in parallel.

Remark. We want to close this section by pointing out two more general concepts that are implicitly covered by our model. Firstly, the TTP model includes the possibility of omitting some holdover edges or even some transport edges of the base network in individual time slots, in order to model restricted operation times of transportation services or hubs. The second concept are abstract aspects of commodities, such as "needs cooling", "is hazardous" and similar features restricting the transportation. These can be modelled by introducing a corresponding property, letting the respective commodities receive a strictly positive extent in this property and accordingly adjusting container capacities.

2.3 Tariff Selection Subproblem

While containers constitute a versatile tool to model various transport tariffs as described in Section 2.2.3, the use of elaborate gadgets significantly increases the number of edges in an instance of our model. Different solution algorithms may or may not be able to cope well with this challenge. In this section, we describe an approach to curb the effects of model blowup due to tariff gadgets by encapsulating tariff selection decisions in a subordinate optimization problem, that we call the *tariff selection* subproblem (TS). While some of our algorithms for TTP introduced in Sections 2.4 and 2.5 will operate directly on tariff gadgets as introduced in Section 2.2.3, others will solve TS repeatedly, possibly very often for each transport relation, while computing a flow pattern for all commodities through the network.

In contrast to the global perspective of the TTP model, TS constitutes a local decision limited to a single transport relation $R \in \mathcal{R}$: Given a fixed vector $\bar{x}(R) \in \mathbb{R}_+^K$ of flow to be transported on R, it asks which transport tariffs should be selected and how should the fixed demand be distributed among selected tariffs in order to meet flow demand at minimum cost? More formally, the problem TS for transport relation $R \in \mathcal{R}$ can be stated as

$$\begin{aligned} \min \quad & \sum_{t \in T(R)} C_t(x(t)) \\ \text{s.t.} \quad & \sum_{t \in T(R)} x_i(t) = \bar{x}_i(R) \quad \forall\, i \in K \\ & x_i(t) \in \mathbb{R}_+ \qquad \forall\, t \in T(R). \end{aligned}$$

A solution to TS comprises a vector $x(t) \in \mathbb{R}_+^K$ of multicommodity flow for each tariff $t \in T(R)$ such that their sum meets the total flow demand $\bar{x}(R)$. From a network-wide perspective, solving the union of the TS problems on all transport relations optimizes transport cost with respect to a given fixed multicommodity flow in the pattern expanded network.

Depending on which of the five types of tariff cost functions introduced in Section 2.2.3 are present in TS, we employ different techniques in order to solve TS. In Section 2.3.1

we devise a mixed integer programming formulation for arbitrary combinations of tariff cost functions in TS. However, out of the different tariff cost functions, property-dependent piecewise constant costs stand out for a number of reasons. First, while they constitute the most elementary class of cost functions, in the presence of multiple tariffs of this type determining an optimal tariff selection already is $\mathcal{NP}$-hard (cf. Proposition 2.1). Second, it may be the tariff type occurring most frequently in logistic applications: Indeed, in the real-life data for our computational study in Section 2.6, many transport relations are equipped exclusively with piecewise constant tariffs. Therefore, Section 2.3.2 is devoted to theoretic and algorithmic insights into TS for this tariff type.

Elaborate algorithms for TTP, which we present later in Section 2.4, solve TS as a subroutine very frequently. Due to its hardness and the demand for extremely short computation times, we rely on fast heuristic algorithms for piecewise constant tariffs yielding only approximate solutions as an alternative to an exact MIP approach. A comprehensive computational study on the performance of such approaches has been published in a separate article by [KMR12]. The findings there outline an elaborate trade-off between solution quality and speed of different greedy algorithms. We close this section by giving a mixed integer programming formulation for the general case and a hardness proof of the special case for piecewise constant costs.

2.3.1 MIP for the General Case

The introduction of tariff gadgets in Section 2.2.4, enables us to naturally formulate and solve TS as a mixed integer program. This versatile approach is especially suited when various tariff types occur together on a single transport relation, or when computational time is not a great issue, e.g., if flow paths for all commodities are already specified and TS only needs be solved once on each transport relation to optimize tariff choice. When each tariff $t \in T(R)$ is represented by a container gadget $(V(t), E(t))$, as detailed in Section 2.2.3, we denote with $E(R) := \bigcup_{t\in T(R)} E(t)$ respectively $V(R) := \bigcup_{t\in T(R)} V(t)$ the set of all edges respectively nodes that are introduced to model the tariff structure on transport relation R. TS for R can then be written as

$$
\begin{aligned}
\min \quad & \sum_{e\in E(R)} c(e)y(e) + \sum_{i\in K} c_i(e)x_i(e) && && \\
\text{s.t.} \quad & \sum_{e\in\delta^+(v)} x_i(e) - \sum_{e\in\delta^-(v)} x_i(e) = \begin{cases} \bar{x}_i(R) & \text{if } v = \text{start}(R) \\ -\bar{x}_i(R) & \text{if } v = \text{end}(R) \\ 0 & \text{otherwise} \end{cases} && \forall\, v \in V(R), \quad \forall\, i \in K \\
& \sum_{i\in K} \alpha_{ij} x_i(e) \le \beta_j(e) y(e) && \forall\, e \in E(R), \quad \forall\, j \in P \\
& y(e) \le u(e) && \forall\, e \in E(R) \\
& y(e) \in \mathbb{Z}_+,\ x_i(e) \in \mathbb{R}_+ && \forall\, e \in E(R), \quad \forall\, i \in K.
\end{aligned}
$$

As this MIP represents TS only on one single transport relation, the MIP instances are rather small and can be solved near-optimally in reasonable time for matters of post-optimization.

2.3.2 Piecewise Constant Costs

When all tariffs on a transport relation are of the property-dependent piecewise constant type, the tariff expanded transport relation is a bundle of parallel fixed-charge container edges. The MIP formulation of TS can be simplified to

$$\begin{aligned}
\min \quad & \sum_{e \in E(R)} c(e) y(e) \\
\text{s.t.} \quad & \sum_{e \in E(R)} x_i(e) = \bar{x}_i(R) && \forall\, i \in K \\
& \sum_{i \in K} \alpha_{ij} x_i(e) \leq \beta_j(e) y(e) && \forall\, e \in E(R),\ \forall\, j \in P \\
& y(e) \leq u(e) && \forall\, e \in E(R) \\
& y(e) \in \mathbb{Z}_+,\ x_i(e) \in \mathbb{R}_+ && \forall\, e \in E(R),\ \forall\, i \in K.
\end{aligned}$$

It is not hard to see that solving TS to optimality remains $\mathcal{NP}$-hard here, even for very restricted special cases. We give a straight-forward reduction from the well-known unbounded knapsack problem, which is proven to be $\mathcal{NP}$-hard [Lue75], to TS instances with only a single property, a single commodity and no bounds on the container multiplicities.

Proposition 2.1 *Problem TS is $\mathcal{NP}$-hard, even when restricted to instances with only piecewise constant cost functions, a single property and a single commodity and unbounded multiplicities.*

Proof. In the single-commodity single-property case, the above MIP reduces to $|E(R)| + 1$ non-trivial constraints, and there remain three single parameters α_{ij}, $\bar{x}_i(R)$ and β_j, which we denote by α, $\bar{x}$ and β, respectively. Every feasible solution satisfies $\alpha\bar{x} \leq \sum_{e \in E(R)} \beta(e) y(e)$, and conversely, if this inequality is satisfied, it is trivial to find feasible assignments $x(e)$. Hence, the MIP reduces in fact to a single non-trivial constraint.

An instance of the unbounded knapsack problem is given by a set of n items with values $v_1, \ldots, v_n \in \mathbb{Z}_+$ and weights $w_1, \ldots, w_n \in \mathbb{Z}_+$, a capacity $W \in \mathbb{Z}_+$ and a desired value $V \in \mathbb{Z}_+$. The task is to find numbers $z_1, \ldots, z_n \in \mathbb{Z}_+$ such that $\sum_{i=1}^n w_i z_i \leq W$ and $\sum_{i=1}^n v_i z_i \geq V$.

Given such an instance I_{UK} of the unbounded knapsack problem, we construct an instance I_{TS} of the above special case of TS as follows. First, for every item $i \in \{1, \ldots, n\}$ of I_{UK}, define $u_i := \lceil W / w_i \rceil$ to be the maximum number of items of type i in a feasible knapsack solution. Then, for each item $i \in \{1, \ldots, n\}$ introduce a corresponding edge e_i with containers of fixed cost $c(e_i) = v_i$ and capacity $\beta(e_i) = w_i$. Moreover, we set $\bar{x} = \sum_{i=1}^n w_i u_i - W$ and $\alpha = 1$.

We now argue that I_{UK} possesses a solution with value at least V if and only if I_{TS} can be solved with cost at most $\sum_{i=1}^n v_i u_i - V$. First assume there is a feasible solution z to I_{UK} with value at least V. We define $y(e_i) := u_i - z_i$ and observe that

$$\sum_{i=1}^n \beta(e_i) y(e_i) = \sum_{i=1}^n w_i (u_i - z_i) \geq \sum_{i=1}^n w_i u_i - W = \alpha\bar{x}$$

and

$$\sum_{i=1}^n c(e_i) y(e_i) = \sum_{i=1}^n v_i (u_i - z_i) \geq \sum_{i=1}^n v_i u_i - V.$$

We omit the converse of the argument as it works analogously. □

2.4 Combinatorial Heuristics

We propose a local search procedure that employs local changes on a path decomposition of flow in the pattern expanded network using tariff selection subroutines. As described in the introduction, there already are a number of local search heuristics available for solving capacitated network design problems. Adapting those methods to multiple capacities and non-binary design variables does not suffice to cope with the large instance sizes occurring from practical application of our model: The precise replication of complex tariff structures leads to a drastically increased number of (mostly parallel) edges, which is further amplified by the cyclic expansion of the network (to give rough numbers, the tariff expanded networks in our computational study have 250,000 edges on average, corresponding to a blow-up factor of 60 from an average of 4000 edges for the base networks). This makes it very hard for heuristics that operate in the tariff expanded network without knowledge of the tariff structure. While most methods known from literature either work directly on the design variables or re-route flow of a single commodity, our approach applies a neighborhood search that is based on path decomposition of flow in the pattern expanded network and re-routes multiple commodities simultaneously.

In order to obtain good initial solutions for the local search algorithm that is presented in Section 2.4.3, we also provide two successive shortest path type algorithms, one that linearizes costs (SPLC) by estimating the per unit cost (Section 2.4.1) and one, denoted by SPTS, that uses a tariff selection method for this purpose (Section 2.4.2). The first method was designed with an emphasis on speed and low memory requirement, while the second is more accurate in cost estimation and is therefore used as the central subroutine in our local search improving moves.

We observed that our local search very well detects cost savings from splitting up flow sharing the same transport relation and re-routing it separately. In contrast, detecting potential savings from consolidating a diverse set of flow carrying paths along a shared subpath is not well captured. Note that this effect may appear only after consolidating multiple paths—identifying such a set of paths is an algorithmically challenging task. In order to address this issue, we adapt the two path-based algorithms to encourage consolidation by (i) forbidding the direct source-sink-connections (which is well-suited for our types of practical networks) in the SPLC heuristic and (ii) using a partial linearization technique for SPTS. Both refinements yield considerable improvements in solution quality of the local search procedure as we will see in Section 2.6.

2.4.1 Shortest Paths with Linearized Costs (SPLC)

A straightforward idea for obtaining shortest path edge weights is estimating the per unit shipping cost on each arc in the tariff-expanded network by linearizing the fixed costs. This technique yields a highly efficient approach suited for solving even the largest occurring instances in a minimal amount of time.

In each iteration, the algorithm chooses a commodity and finds a shortest path from a source to a sink. Whenever the algorithm encounters an edge during the shortest path computation, the (residual) capacity for the chosen commodity on this edge is computed and the fixed cost for that edge is divided by this capacity to obtain a linear cost rate. To make this more precise, let k be the commodity that is currently being routed and (x, y) be the

current (partial) solution. For arc $e \in E$, we compute the *residual capacity for commodity* k:

$$\rho(e) = \min_j \frac{\beta_j(e)y(e) - \sum_i \alpha_{ij}x_i(e)}{\alpha_{kj}}.$$

If there is residual capacity $\rho(e) > 0$, we set the edge capacity $r(e) = \rho(e)$ and the edge weight $w(e) = c_k(e)$. If no residual capacity is left ($\rho(e) = 0$) another copy of this container can be selected if $y(e) < u(e)$. In this case we set

$$r(e) = \min_j \frac{\beta_j(e)}{\alpha_{kj}} \quad \text{and} \quad w(e) = c_k(e) + \frac{c(e)}{r(e)}.$$

Otherwise, if $y(e) = u(e)$, we set $r(e) = 0$ and $w(e) = \infty$.

Once a shortest path from a source to a sink of commodity k w.r.t. the weights w is found, a maximum amount of flow of commodity k w.r.t. the bottleneck of edge capacities r on this paths is sent. Note that all computations above can be carried out very efficiently and of course, instead of updating weights and capacities of all edges in each step, these are calculated on-demand and only invalidated when neccessary.

The linearization procedure assumes optimal utilization of container capacities in the resulting flow pattern and thus favors large containers with low per unit cost rates. Since this high utilization is not always attained, the linearization leads to suboptimal tariff choices on transport relations. The effect can be compensated by optimizing the tariff selection on each transport relation a posteriori with a tariff selection method described in Section 2.3.

Algorithm 2.1: Successive shortest path algorithm with linearized costs (SPLC)

```
1 Initialize x = 0, y = 0.
2 for each commodity i ∈ K do
3     Invalidate r(e) and w(e) for all e ∈ E.
4     while there is a source s of i with remaining supply do
5         Find path P in G from s to a sink t with Σ_{e∈P} w(e) minimum.
6         Augment x along P by min_{e∈P} r(e) units of commodity i, adjust y accordingly.
7         Invalidate r(e) and w(e) for all e ∈ P.
```

Consolidation by forbidding direct connections (SPLC-F)

The SPLC heuristic favors large containers with low per unit cost rates and prefers direct connections as single detours cannot yield lower per unit cost. A simple approach for encouraging consolidation when costs are just linearized is to forbid all direct connections between sources and sinks of the same commodity during the construction of the initial solution. By doing so, hubs and common paths are automatically used. Unnecessary detours can be easily identified and corrected by improving moves of the local search procedure.

2.4.2 Shortest Paths with Tariff Selection (SPTS)

The rather imprecise estimation of the actual transportation cost achieved by the linearization approach presented in the previous section might lead to weak choices of paths when routing

the commodities. We thus propose a second strategy that employs tariff selection algorithms already during the shortest path search. Although this more sophisticated approach requires more computational effort, it still turns out to be very efficient while at the same time providing several possibilities for adjustments.

Since tariff selection methods require as input the amount of flow to be routed, these flow values $\Delta \in \mathbb{R}_+^K$ have to be determined *before* the shortest paths computation. We implement this a priori flow computation efficiently by identifying source-sink-pairs such that the possible transport volume from source to sink is maximum (w.r.t. a weighted combination of the property extents).

More formally, for each ordered pair of nodes (s,t) in the pattern expanded network, let

$$\Delta_k(s,t) := (\min\{b_k(s), -b_k(t)\})^+$$

for $k \in K$, and let $w \in \mathbb{R}_+^P$ be the weight function, given as a parameter to the heuristic. Then source s and sink t are chosen such that $\sum_{j\in P} w_j \alpha_j(\Delta(s,t))$ is maximum.

During the shortest path computation, arc weights have to be evaluated too often to solve the tariff selection problem to optimality every time. In fact, it is sufficient to only estimate the cost while the actual tariff assignment can be determined at the end of the solution process from the flow values on the transport relations in the pattern expanded network using an exact method. For the cost estimation we use the covering relaxation from [KMR12, Section 3.5].

Algorithm 2.2: Successive shortest path algorithm with tariff selection (SPTS)

1 Initialize $x = 0$.
2 **while** *not all demand has been satisfied* **do**
3 Let $s, t \in \mathcal{V}$ such that $\sum_{j\in P} w_j \alpha_j(\Delta(s,t))$ is maximum.
4 Compute shortest path P in $\mathcal{G}$ from s to t w.r.t. $\tilde{c}$, where $\tilde{c}(R)$ is the estimated cost for augmenting the current flow $x(R)$ by $\Delta(s,t)$ on transport relation R.
5 Augment x along P by $\Delta(s,t)$.
6 Compute a flow in the tariff expanded network of same value as x using a tariff selection method.

Consolidation by partial cost linearization (SPTS-L)

Cost computation based on tariff selection allows for a more sophisticated approach to encourage consolidation by taking into account the unrouted demand. We linearize costs at inter-hub and source-hub (if there are fewer sources) or hub-sink (if there are fewer sinks) connections in the following way: Let $\Delta^+ \in \mathbb{R}_+^K$ be the sum of all supply not yet routed in the current solution, and let $M := \min_j \frac{\sum_i \alpha_{ij}\Delta_i^+}{\sum_i \alpha_{ij}\Delta_i}$. For each available tariff t on a transport relation, we now compute the cost $C_t(\Delta^+)$ for routing Δ^+ and divide it by M to obtain an edge cost that anticipates future consolidation on this transport relation.

2.4.3 Path-based Local Search

In the following we introduce a local search algorithm that re-routes flow along paths with the aim of improving feasible solutions. Before we describe the procedure in detail, we shortly introduce the notion of flow decomposition.

A well-known result from network flow theory states that any feasible flow in a network can be decomposed into flow on paths from sources to sinks (and cycles, which however can immediately be removed from the solution in our case). A *flow-carrying path* is a tuple (P, Δ_P), where P is a sequence of transport relations $R_1, \dots, R_m$ such that $\text{start}(R_{i+1}) = \text{end}(R_i)$ and $\Delta_P \in \mathbb{R}_+^K$ is a multicommodity flow vector specifying the amount of flow sent along the path. A *path decomposition* of a flow x is a collection of flow-carrying paths $\mathcal{P}$ such that $x(R) = \sum_{P \in \mathcal{P}: R \in P} \Delta_P$.

The local search algorithm maintains a path decomposition of the flow of the current solution. It moves from one solution to another by replacing one or multiple paths of the decomposition with paths of lower cost. The general outline of an improving move is the following: When removing a path (P, Δ_P) from the solution, for each transport relation R of the path, $x(R)$ is decreased by Δ_P and the tariff selection of R is adapted accordingly, using the greedy tariff selection heuristic presented in Section 2.3. After removing a set of paths, the resulting partial solution is completed again by computing new paths using the SPTS heuristic introduced in Section 2.4.2. The move is accepted if the total cost of the solution decreases, and reverted otherwise.

We implemented two variants of improving moves: Type A moves simply remove a single path at a time. This way, only small amounts of flow are re-routed in one move and the assignment of sources to sinks is left unaffected. In contrast, Type B moves consider groups of paths sharing the same transport relation. All flow passing this transport relation is removed and routed anew, which means that multiple paths can be replaced at once and the assignment of sources to sinks might be altered.

Our local search algorithm now performs improving moves in alternating phases of Type A and B. This allows us to re-compute the path decomposition at the beginning of each phase, adapted to the type of movement.

In both cases paths are constructed in a DFS manner: At a node in the DFS tree for each incident edge R we compute the maximal flow vector $\Delta(R)$ that could be assigned to a path proceeding on that edge and choose an edge greedily so as to maximize a suitably defined weight function of that flow vector. For Type A phases, the DFS starts at a source and continues along the edge that maximizes a weighted combination of the properties of $\Delta(R)$. In contrast, the decomposition for Type B phases facilitates a bidirectional DFS starting at heavily used transport relations and chooses edges that maximize the savings resulting from reducing their flow. In both cases, due to flow conservation we either close cycles (which can immediately be removed from the solution) or find a source-sink path, which we add to the path decomposition.

The two phases are repeated alternatingly until the relative improvement achieved by both of them falls below a specified value or the time limit is reached. At the end of the procedure, a final improvement phase is conducted by identifying and eliminating weakly utilized containers in the tariff expanded network and again re-routing the corresponding flow using a variation of type B moves.

2.5 MIP-Based Approaches

In this section, we discuss mixed integer programming techniques that supplement the combinatorial heuristics presented in the previous section, not only yielding high quality solutions but also providing lower bounds for assessing this quality. The plain MIP formulation presented in Section 2.2.4 is not suited for solving reasonably sized real-world networks since they involve too many variables and constraints. We propose an aggregated formulation that considerably reduces model size and still yields good dual bounds (Section 2.5.1). We then combine this with efficient preprocessing techniques to tighten the relaxation (Section 2.5.2). In Section 2.5.3, we use solutions to the LP relaxation of this strengthened aggregated formulation as initial solutions for our local search. Finally, a post-processing step that improves solution quality is presented in Section 2.5.4. During this post-processing step, tariff selection decisions are locally optimized on all transport relations that connect a given pair of nodes in different slots of the pattern expanded network.

Besides strengthening the MIP formulation, a promising approach to deal with multicommodity capacitated network flow problems is to use a Benders Decomposition, see e.g., [CCG09]. Preliminary runs with a Benders Decomposition combined with heuristics and adding additionally valid inequalties implemented in SCIP version 2.0 suffered from slow solving times. Interestingly, the subproblems (multi-commodity multi-dimensional flow problems) solved by CPLEX turned out to be the bottleneck. In fact, numerical instability results from high variance between large and small coefficients in our practical instances in conjunction with inexact dual values inherent to Benders decomposition. Experiments with warm starts in the subproblem solving procedure and other techniques did not work out on our large scale tariff and pattern expanded networks. To be precise, CPLEX tries several Markowitz thresholds and tries to repair basis singularities. We point out that on small sized instances our Benders implementation works well but it seems to be the large instances that induce a huge amount of Benders cuts together with their widely varying coefficients and long LP solving times that make the difference here. We leave it to future research on how to incorporate multi-dimensional capacities into combinatorial approaches similar to [CCG09].

2.5.1 Tariff Aggregated MIP (AMIP)

As mentioned above, the plain MIP formulation suffers from huge memory requirements. In particular the introduction of tariff gadgets results in a tremendous number of—mostly parallel—edges. We make use of this parallel structure and propose an aggregated formulation that still reflects the original tariff structures while significantly reducing the number of flow variables and capacity constraints. The aggregation is set up as follows. For each pair of nodes $v, w \in V$ let $E(v, w)$ be the set of edges from v to w in the tariff expanded network. For each $i \in K$, we replace the flow variables $x_i(e)$ of the edges $e \in E(v, w)$ by a single flow variable $x_i(v, w) \in \mathbb{R}^K_+$. For each $j \in P$, we replace the capacity constraints of the edges in $E(v, w)$ w.r.t. j by a single constraint

$$\sum_{i \in K} \alpha_{ij} x_i(v, w) \leq \sum_{e \in E(v,w)} \beta_j(e) y(e).$$

Clearly, the resulting MIP is a relaxation of the original TTP instance, as we can construct a feasible solution of the relaxation from a feasible solution of the original formulation by setting

$x_i(v,w) := \sum_{e \in E(v,w)} x_i(e)$ and adopting the values of all design variables. Conversely, each solution of the relaxation induces a flow on the transport relations of the pattern expanded network. These flow values yield a tariff selection subproblem on each transport relation (see Section 2.3). Computational experiments on practical instances reveal that by applying a tariff selection heuristic on each relation, we can derive feasible solutions of the original model with only a minimal increase in cost. On the other hand, given the typically high number of parallel edges between each pair of nodes in TTP instances (20 on average in our test sets), the aggregation drastically reduces the number of variables and constraints, resulting in a considerable boost in effectiveness of branch and bound solvers.

2.5.2 Preprocessing

Although tariff aggregation helps to reduce problem sizes, the considered MIPs still suffer from numeric instability and weak lower bounds. We address these issues in the following paragraphs with two preprocessing steps that can be applied to the aggregated formulation.

Strengthened container inequalities

As already discussed in Section 2.1.3, MIP formulations of capacitated network design problems can be considerably strengthened by adding valid inequalities. Among the valid inequalities used in literature are strong capacity and minimum cardinality inequalities. The natural extensions of these inequalities to TTP, however, did not turn out to be very effective for the instances in our computational study. Instead, we propose a method to bound the total extent of capacity used within individual containers. Before we describe these *strengthened container inequalities* in detail, we give some reasons for the failure of the known inequalities mentioned above.

Strong capacity inequalities state that $x_i(e) \leq b_i y(e)$ for all $i \in K$ and all $e \in E$, where $b_i := \sum_{v \in V : b_i(v) > 0} b_i(v)$ is the total demand of commodity i. While [CCG11] report on the positive impact of strong capacity inequalities on the integrality gap in their computational experiments, it is also easy to see that the strong capacity inequality for commodity i at edge e can only strengthen the original formulation if $\beta_j(e) > \alpha_{ij} b_i$ for all $j \in P$. In typical TTP instances arising in practice, total demands within the network are much larger than individual transport capacities and the inequalities remained mostly ineffective.

Minimum cardinality inequalities require the number of containers installed on a cut induced by a set of nodes $S \subset V$ to be at least as large as the minimum number of containers required to transport the excessive demand $(\sum_{v \in S} b_i(v))^+_{i \in K}$ within S across the cut. As already observed by [CCG11], these inequalities are weak if the magnitudes of the capacities vary widely, as it is typically the case for logistics tariffs that are modelled within TTP instances. Their suggested improvements cannot be applied in our case as their model contains only binary design variables whereas ours are integer. In the following, however, we show how to strengthen our capacity inequalities using similar ideas.

Solutions to the linear programming relaxation of TTP provide weak lower bounds for the following reason: When considering a flow carrying transport relation, LP solutions tend to set the variable of the largest container to the minimal fraction needed to grant capacities for the flow on this transport relation. These fractions are unfortunately very small, which means that they do not reflect the cost that would be incurred in an integer solution. The

idea is to restrict container capacities without affecting the cost of an optimal integer solution. This is possible, if for a given transport relation $R \in \mathcal{R}$ an upper bound $\Gamma(R)$ on the flow $x(R)$ in any optimal solution is known. Useful upper bounds can be derived for transport relations incident to node sets $S \subset V$ with either $\delta^+(S) = \emptyset$ or $\delta^-(S) = \emptyset$. Given an upper bound $\Gamma(R)$, we can replace for every $e \in E(R)$ and every $j \in P$ the capacity $\beta_j(e)$ by $\beta_j(e) - s_j$, where s_j is the result of solving

$$\begin{aligned} \min \quad & s_j & & \\ \text{s.t.} \quad & \sum_{i \in K} \alpha_{ij'} x_i(e) + s_{j'} &= \beta_{j'}(e) \qquad & \forall\, j' \in P \\ & 0 \le x_i(e) &\le \Gamma_i(R) \qquad & \forall\, i \in K \\ & s_{j'} &\ge 0 \qquad & \forall\, j' \in P. \end{aligned}$$

In a preprocessing routine we solve these linear programs for each property j of each fixed charge container e on each transport relation R for which reasonable upper bounds $\Gamma(R)$ can be computed.

Commodity scaling

We could observe numerical difficulties while solving LP relaxations of large instances: The LP solving steps suffer from basis singularities and sometimes even numerical infeasibility. One reason for these difficulties lies in the diversity of properties for different commodities. The capacity inequalities involve many flow variables with property coefficients varying in magnitudes of 10^6 for our test instances. Nonetheless, because flow variables are fractional in our model, we can apply the following scaling steps. For each commodity $i \in K$ we determine a scaling factor $s_i > 0$ and obtain scaled values $\tilde{b}_i(v)$ and $\tilde{\alpha}_{ij}$, defined by

$$\tilde{b}_i(v) := b_i(v)/s_i \quad \text{and} \quad \tilde{\alpha}_{ij} := s_i \alpha_{ij} \ \text{ for each } j \in P.$$

The scaled problem instance is equivalent to the non-scaled one in the sense that feasible flow values $\tilde{x}_i(e)$ obtained for the scaled problem can be scaled back to obtain feasible flow values $x_i(e) = s_i \tilde{x}_i(e)$ for the original problem. We chose the scaling factors s_i for each commodity in such a way, that among the resulting coefficients $\tilde{\alpha}_{ij}$, $j \in P$ the smallest such coefficient has the magnitude 10^{-1}. The improved numeric stability of the constraint system significantly speeds up the LP solution process.

2.5.3 Initial Solutions for Local Search from Aggregated LP Relaxation (ALP)

In Section 2.4, we discussed the importance of properly chosen initial solutions for the local search procedure, and devised two ways to encourage consolidation of flow during the construction of the initial solution by shortest path type algorithms. Alternatively, we can obtain initial solutions from the LP relaxation of the aggregated MIP formulation by applying tariff selection heuristics to the multicommodity flow in the pattern expanded network induced by the aggregated LP solution.

Notice that in this case, strengthening container inequalities as described above also encourages consolidation in the solution process. In fact, the effect of the strengthened inequalities is strongest on edges that are reachable from few sources or sinks only (such as

direct source-sink connections). This implicitly encourages flow to take detours on longer source-sink paths, where less strong container inequalities permit lower costs in the LP relaxation. Since inappropriately consolidated flow can be efficiently disaggregated by the local search algorithm, initial solutions constructed from the LP relaxation lead to high quality final solutions as we shall see in Section 2.6.

2.5.4 Pattern Optimization Subproblem

In the tariff selection subproblem considered in Section 2.3, we fixed the amount of flow passing a given transport relation and optimized the tariff selection with respect to this given flow value. This idea can be extended by considering all transport relations that connect a given pair of nodes in different slots of the pattern expanded network. More formally, for some node $v \in B$ in the base network and a cycle length F, let $v_1, \dots, v_F$ be the copies of node v created in the pattern expansion step, with $v_i \in V(B_i)$ for $i \in \{1, \dots, F\}$. We consider the *Pattern Optimization Subproblem* induced by a fixed pair of nodes s, $t \in B$ and therefore define

$$V(s,t) := \bigcup_{i=1}^{F} \{s_i, t_i\}, \qquad \mathcal{R}(s,t) := \{R \in \mathcal{R} : \text{start}(R), \text{end}(R) \in V(s,t)\}.$$

Given a solution to the whole TTP instance with flow values $(\bar{x}(R))$, $R \in \mathcal{R}$, we consider a locally restricted instance of TTP, fixing the flow values on all transport relations $\mathcal{R} \setminus \mathcal{R}(s,t)$ and optimizing the flow $(x(R))_{R \in \mathcal{R}(s,t)}$ in the subnetwork induced by the copies of s and t, i.e.,

$$\begin{aligned}
\min \quad & \sum_{R \in \mathcal{R}(s,t)} \sum_{t \in T(R)} C_t(x(t)) \\
\text{s.t.} \quad & \sum_{R \in \delta^+_{\mathcal{R}(s,t)}(v)} x_i(R) - \sum_{R \in \delta^-_{\mathcal{R}(s,t)}(v)} x_i(R) = \bar{b}_i(v) && \forall\, v \in \mathcal{V}(s,t),\ \forall\, i \in K \\
& \sum_{t \in T(R)} x_i(t) = x_i(R) && \forall\, R \in \mathcal{R}(s,t),\ \forall\, i \in K \\
& x(t) \geq 0 && \forall\, t \in T(R),\ \forall\, R \in \mathcal{R}(s,t)
\end{aligned}$$

where $\bar{b}(v) := \sum_{R \in \delta^+_{\mathcal{R}(s,t)}(v)} \bar{x}(R) - \sum_{R \in \delta^-_{\mathcal{R}(s,t)}(v)} \bar{x}(R)$. Using tariff gadgets, this restricted instance of TTP can be formulated as a mixed integer program. It contains only a small fraction of the decision variables present in the whole instance. In fact, restricted instances can be solved to near-optimality very quickly using a standard MIP solver. We thus iteratively optimize these subproblems arising for all pairs of adjacent nodes with flow carrying transport relations in between them.

Note that in contrast to the tariff selection subproblem, solving the pattern optimization subproblem for one pair of nodes may affect the subproblem of other, non-disjoint pairs of nodes, as holdover edges of a common node appear in each of the problem as variables. Consequently, the order of the node pairs considered plays an important role. We order the node pairs non-increasingly with respect to a weighted combination of the property extents of the total flow in the subnetwork affected by the pattern optimization of each pair (s, t), i.e., $\sum_{j \in P} w_j \alpha_j(\sum_{R \in \mathcal{R}(s,t)} \bar{x}(R))$, using the same weights $w \in \mathbb{R}^P_+$ as provided for local search and

SPTS heuristic. This reflects the optimization potential of the corresponding node pair and leads to an "important pairs first" order, which is also useful when the pattern optimization process is not carried out on all node pairs due to time constraints.

2.6 Computational Study

We verify the TTP model and the algorithmic approaches presented in the preceding sections by conducting a computational study based on real-world data provided by our project partner 4flow AG, a logistics consultancy company serving small, medium-sized and global customers from a broad spectrum of industries. We also compare our heuristics and MIP based approaches with a reference solution obtained from 4flow AG.

2.6.1 Instance Sets

The benchmark library consists of 145 instances aggregated from four recent and on-going customer projects in three different industries (Auto1, Auto2, Chemical, Retail). All base networks correspond to European supply chains in which goods are transported according to full truck load (FTL) or less than truck load (LTL) tariffs. These networks share a layered graph structure. More specifically, the nodes of the base network are partitioned into an ordered set of layers, with the lowest layer containing all sources, and the highest layer containing all sinks. In addition, there is a fixed number (varying from one to three) of intermediate hub layers. There is a transport relation between every pair of nodes from distinct layers, directed towards the higher layer. However, transport relations within the same layer are not present. Pattern expansion has been conducted with a cycle length of six slots—one slot corresponds to two months of a year. All tariffs are of piecewise constant type, depending on the same two properties (mass and volume) in every instance.

While the automotive instances represent production networks with a high number of sources and a low number of sinks, the chemical industry and retail sets are based on distribution networks with a high number of sinks but only few sources. Table 2.2 shows the average values of key parameters of the instances within each set: the first three columns contain the number of sources, sinks, and hubs in the base network, followed by the number of commodities (comm.), and the number of edges in the base network, pattern expanded network and tariff expanded network.

set	#nodes in base network			#comm.	#edges		
(#instances)	#source	#sinks	#hubs		base	pattern exp.	tariff exp.
Auto1 (36)	35	6	7	162	335	2296	76653
Auto2 (18)	34	3	4	117	186	1364	29264
Chemical (50)	7	244	19	101	6601	41222	239238
Retail (41)	4	177	26	307	5665	35229	511064

Table 2.2: Average sizes of the instances per set

For future research, the instance library will be available upon request after signing a contract of data confidentiality. For more information, please contact one of the authors.

2.6.2 Algorithms and Implementation Details

We implemented and tested different variants of the algorithms presented in Sections 2.4 and 2.5 in order to determine good parameter settings and combinations. In long term planning, running time plays a minor role and the fine-tuned aggregated MIP formulation combined with the path-based local search and pattern optimization with generous time limits can be used. In order to enable the evaluation of multiple scenarios, our industrial partner set a time limit of 30 min. For this case, we also provide test results of approaches designed for time-efficiency without sacrificing too much solution quality.

Overall, the following algorithms were tested on all 145 instances of the benchmark library. The first two algorithms correspond to MIP approaches and the last four are local search procedures that are named according to the algorithm that delivers the initial solution for local search.

`AMIP-H` Aggregated MIP with integrated local search (cf. Section 2.6.2);

`MIP` Plain MIP formulation for comparison purposes, see Section 2.2;

`ALP` LP relaxation of aggregated MIP formulation (cf. Section 2.5.3)

`SPLC` Shortest path heuristic with linearized cost (cf. Section 2.4.1);

`SPLC-F` same as `SPLC`, but with forbidden direct connections (cf. Section 2.4.2);

`SPTS-L` Shortest path heuristic with tariff selection (cost estimator) and partial linearization (cf. Section 2.4.2);

All algorithms have been implemented in C++ and compiled with gcc 4.5.0 on openSUSE 11.3 Linux with kernel 2.6.32.19-0.2. Computations have been performed on cluster nodes with two DualCore-Opteron 2218 processors (2.6 GHz, 64 bit) and 16 GB of memory using CPLEX 12.1 for MIPs and LPs. Since the heuristic approaches have not been adapted to support concurrency, we limited the number of threads for the CPLEX solver to one to ensure comparability of the results.

In the following Section 2.6.2 we elaborate on the interplay of the MIP and the local search heuristic, whereas the detailed settings for the variants of local search procedures are presented in Section 2.6.2.

Branch and Bound Frameworks.

Our tests involved different MIP formulations, which we implemented in CPLEX. We tested the plain MIP formulation (`MIP`) for a direct comparison with our algorithms as well as the aggregated MIP formulation that includes the preprocessing methods described in Section 2.5.2 and callbacks to our heuristics. The resulting algorithm is denoted by `AMIP-H` and details of the implementation are given below. In order to obtain reasonably tight lower bounds, we also ran the aggregated MIP formulation without heuristic callbacks (`AMIP-B`). We invoked a time limit of 2 h for the branch and bound process, and an extra time of 1 h for applying local search and pattern optimization each.

When solving the aggregated and preprocessed MIP formulation from Section 2.5 with a branch and bound framework, we apply the local search and pattern optimization procedures

throughout search on integer solutions as well as fractional LP solutions obtained in a node of the branch and bound tree. These solutions induce a flow on the transport relations of the pattern expanded network. This flow can be turned into a feasible TTP solution by solving the tariff selection subproblem on each transport relation (cf. Section 2.5.1). We further improve this solution by applying the local search heuristic and pattern optimization with a time limit of 300 s.

As this procedure incurs a significant computational effort, we require at least 1,500 branch and bound nodes to be processed between two successive calls of the heuristics. Furthermore, we use the cost estimation from the covering relaxation in [KMR12, Section 3.5] in order to evaluate the potential of a given LP solution to improve on the currently best solution: Only if the estimated total cost is within 8% to the best known solution, we compute the corresponding TTP solution. We also apply the procedure to all integer solutions found by the MIP solver.

Local Search Procedures

We tested the local search algorithm described in Section 2.4.3 using initial solutions constructed by the aforementioned heuristics. The current tariff selection on the transport relations is then further improved using the exact MIP formulation as described in Section 2.3. Finally, pattern-optimization is performed on the returned solution using the non-aggregated formulation.

Computation time of the starting heuristics `ALP`, `SPLC` and `SPLC-F` turned out to be almost negligible, and we invoked a total solution time of 30 min (including pattern optimization) in this case. Unfortunately, the more sophisticated `SPTS-L` solver turned out to cause considerably more computational effort. Here we invoked the same time limits as for the branch and bound approaches. Recall that for fine-tuning the path-decomposition of the local search procedure and the `SPTS` heuristic an additional parameter is specified—a weight function on the properties of the model that reflects the importance of properties. For the benchmark instance set, mass occurs to be the dominant property. We thus choose the weight function to be an indicator function on mass.

2.6.3 Results

We now elaborate on the results of our computational experiments, starting with the effect of aggregation on the lower bounds. We then analyze solution quality and the impact of local search initial solutions and pattern optimization. We close by comparing our approach to a reference solution on an additional instance.

Influence of Aggregation on Lower Bounds

We investigate the improvement on lower bounds achieved by the aggregation and our preprocessing techniques against the plain MIP formulation in Table 2.3. In fact, we observed that especially for the large instances, `MIP` suffers from numerical instabilities and degeneracy that lead to solving times of thousands of seconds for the root relaxation. In some cases, the initial cut generation rounds for the root relaxation did not terminate within given time limits. In turn, the efficiency of initial cuts greatly benefits from our preprocessing techniques—fewer cuts achieve a much better lower bound here.

solver	Auto1 (31/36)	Auto2 (18/18)	Chemical (48/50)	Retail (30/41)	all (127/145)
ALP	-17.81	-6.87	-21.61	-10.44	-15.96
AMIP-B	17.48	0.61	12.38	4.84	10.18

Table 2.3: Influence of Aggregation on Lower Bounds Average improvement of the lower bound compared to MIP. The number of instances handled by MIP/number of all instances per set is shown in parentheses.

solver	Auto1 (36)	Auto2 (18)	Chemical (50)	Retail (41)	all (145)
MIP	9.09 (1)	2.35 (3)	29.18 (0)	13.33 (1)	16.38 (5)
ALP	6.22 (24)	2.63 (0)	13.92 (17)	4.74 (40)	8.01 (81)
AMIP-H	6.12 (26)	1.26 (16)	14.61 (24)	4.75 (38)	8.06 (104)
SPLC	6.51 (17)	5.07 (0)	23.54 (0)	4.75 (37)	11.71 (54)
SPLC-F	6.90 (11)	3.65 (0)	18.08 (9)	10.70 (27)	11.43 (47)
SPTS-L	6.57 (19)	4.15 (0)	19.44 (1)	4.74 (39)	10.19 (59)

Table 2.4: Average gaps to best known lower bound in % The number of achieved best solutions is shown in parentheses.

Not surprisingly, the lower bounds derived by the strengthened aggregated LP (ALP) are of low quality, with a gap of more than 15% on average towards the value obtained by MIP. In a set-by-set comparison, the AMIP-B method achieves an average improvement over MIP of more than 10%, and of up to 17% on average on set Auto1, while MIP is only competitive on the comparatively small instances of the Auto2 set. Apparently, the loss in tightness caused by the aggregation is more than compensated by the boost in efficiency of the branch and bound procedure achieved by the smaller size of the formulation and its increased numerical stability. An overview over the lower bounds for all instances can be found in the Appendix.

Quality of Solutions

Table 2.4 shows the gaps of the computed solutions to the lower bound computed by AMIP-B. Throughout the automotive and retail instance sets, the solution quality is within single-digit average gaps to the lower bounds. The local search with LP starting solution and the AMIP-H framework provide the best solution quality, while the performace of approaches with path-based initial solutions is weaker and varies depending on the instance set. We infer that the more holistic LP approach captures the multi-commodity flow nature of our problem better than the iterative path approaches.

AMIP-H attains near-optimality on Auto2, outperforming ALP on this set. Apparently, the small instance sizes in this set benefit the branch and bound process. The gaps are considerably weaker on the instances of the Chemical set. The instances of this set are much bigger w.r.t. the number of edges and sinks in the base network than those from the other sets, which presumably also affects the MIP framework's ability to produce tight lower bounds.

Performance of Local Search and Impact of Initial Solutions

The results in Table 2.4 and Figure 2.2 show that the choice of the initial solution clearly affects the performance of the local search procedure. In fact, on many instances, the initially expensive flow patterns of the consolidation enforcing heuristics lead to better final solutions

than those obtained from solutions with low consolidation provided by SPLC for comparison. However, the effectiveness of the combinatorial starting heuristics strongly depends on the structure and size of the instance. In contrast, ALP consistently shows best results, on par with the AMIP-H framework (which takes considerably more computational effort).

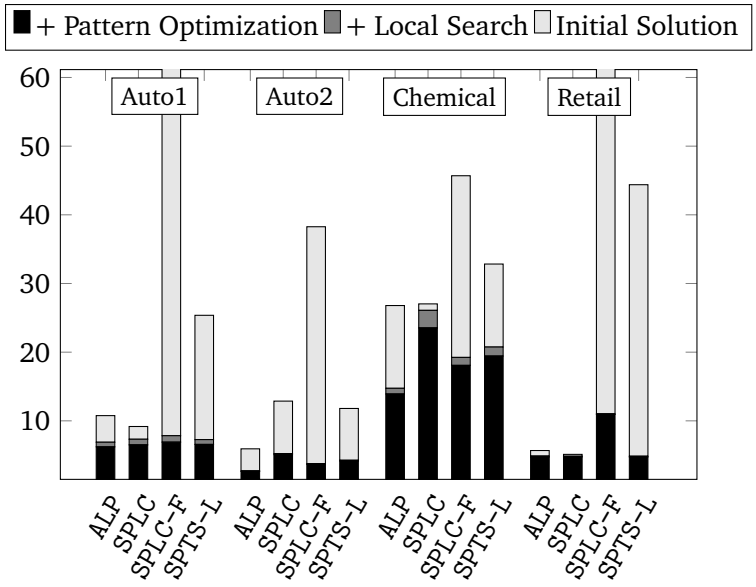


Figure 2.2: Gaps Achieved with Postprocessing in %. The percentaged gap to best known lower bound of fast solvers (time limit 1800 s) for the initial solution, after local search and after pattern optimization are shown—with initial solutions by SPLC-F achieving 113% in average on Auto1 and 253% on Retail

Impact of Pattern Optimization

Figure 2.2 reveals that the effect of pattern optimization is rather weak on the sets Auto2 and Retail, while its share of the computation times is significant (Figure 2.3). The picture is considerably different, however, for the instances of the Chemical set. Here, computation times are reduced to a minimum, while the improvement of solution quality due to pattern optimization is significantly higher. This better performance can be explained by the less granular tariff structure in this instance set, resulting in smaller subproblems while at the same time increasing the importance of temporal consolidation.

Purely Combinatorial Heuristics

In order to provide solutions independent of third party software and licenses, we also evaluated purely combinatorial variants of the local search heuristic with path-based initial solutions: After replacing MIP based tariff selection algorithms with greedy heuristic and omitting pattern optimization, the approaches still produce good solutions with a mild loss of at most 3% points of average solution quality.

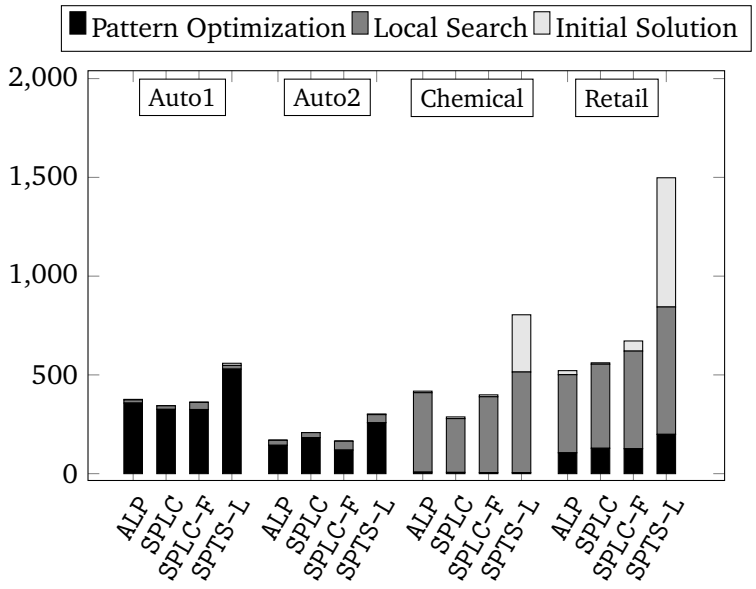


Figure 2.3: Running Times of Postprocessing. The table shows shifted geometric means of running times as shares of 2000 s for thelocal search combined with different algorithms to compute initial solutions and with pattern optimization.

Comparison with Solutions from Practice

Due to confidentiality reasons we could not obtain reference solutions or current network costs for the instances presented above. Instead, a direct comparison with an instance of a European cross-docking network from a recent customer project has been conducted in cooperation with 4flow AG. The base network consists of 228 consumers, 545 suppliers, 5 hubs, 5857 edges, resulting in a tariff expanded network with 209304 edges. It is fully connected in contrast to the layered structure observed so far. On this instance, the `AMIP-H` framework obtained a solution with 1.2% gap to optimality. We compared this against a solution obtained with a standard software for supply chain design at project start operating on a conventional model. Our solution constitutes a 14% improvement, which, if applied on an annual basis, results in savings of up to 1.6 million Euro.

2.7 Summary & Conclusions

The tactical transportation planning model presented in this chapter integrates the important aspects of tactical logistics network optimization: realistic transportation tariffs, delivery patterns, and inventory costs. Several algorithmic techniques have been devised to address the challenges associated with the specific instance structure. These methods have been successfully tested on a broad set of real-world instances.

Among our techniques, we propose a local search procedure that simultaneously re-routes flow of multiple commodities. Equipping the local search with different types of initial solutions, such as multicommodity flow patterns derived from a strengthened LP relaxation or from purely combinatorial path-based approaches, yields solutions that are within a single

digit percent of the optimum on average. Our algorithm can be used both in connection with standard MIP solvers for optimal solution quality, or as purely combinatorial algorithm, yielding competitive solutions without usage of third-party software. Hence, the broad spectrum of our algorithms offers a flexible trade-off between solution quality, operating cost and computation time.

The performance of our algorithms to a great part relies on the successful isolation of the tariff selection subproblem. We devise a variety of exact and heuristic methods to efficiently solve this problem, providing a tradeoff between speed and exactness of the solution procedure. A computational analysis of these algorithms and additional techniques can be found in a companion paper to this article by König, Matuschke, and Richter [KMR12] that also provides further theoretical insights into the tariff selection subproblem.

Chapter 3

Special Cases and Tractability

3.1 Introduction

Motivated by the model from Chapter 2 for tactical transportation planning, we are interested in the problem variants on general graphs, where we assume that commodities cannot be split into different routes. We focus on special cases and their tractability. Note that this unsplittable requirement may render otherwise simple multicommodity flow problems hard. For example, the min-cost single source unsplittable flow problem is $\mathcal{NP}$ -hard: Commodities must be routed from a single source node to destination nodes respecting edge capacities and each commodity must be routed along a single path. We refer to [Sku00] for hardness proofs and approximation algorithms for several variants of this problem. Note that these problems have edge capacities which does not apply for logistics networks in the hard sense, i.e. transportation capacities can always be obtained by a freight market. The "capacities" in the model of Chapter 2 are only a result from modeling tariff systems.

We thereby focus on the multidimensional properties of commodities and resulting consolidation effects. In this regard, we assume linear transportation costs and ignore tariff granularity and economy-of-scale effects. In its most general form, allowing more properties than weight and volume, the problem is:

Problem 1 (CONSOLIDATION STEINER SUBGRAPH)

Given: A directed graph $G = (V, E)$; a set of properties Π; a set of demands $J = \{D_1, \dots, D_k\}$ with $D_j = (\text{source}_j, \text{sink}_j, d_j)$, where source_j and sink_j are source and sink vertices, respectively, and $d_j : \Pi \to \mathbb{R}_+$ describes the properties of the demand; a cost function $c : E \times \Pi \to \mathbb{R}_+$.

Goal: Find a list of paths $P_1, \dots, P_k$ in G, where P_j connects source_j with sink_j and their cost is minimal:

$$\sum_{e \in E} \max_{\pi \in \Pi} \left(c(e, \pi) \sum_{j : e \in P_j} d_j(\pi) \right). \tag{3.1}$$

In Problem 1 we pay for each arc only the cost incurred by the most expensive property. We can think of the unit of $d_j(\pi)$ as e. g. kg or m^3 and the unit of $c(e, \pi)$ as \$/kg or \$/m^3.

Observation. The paths $P_1, \ldots, P_k$ can w.l.o.g. be assumed to be simple because canceling a cycle in some path P_j does not increase cost. □

Lemma 3.1 *If we drop the "unsplittable" restriction and allow a fractional multicommodity flow, that is, each demand may be satisfied by multiple paths P and a value $0 \leq r_j^P \leq 1$ determines the fraction of d_j to be satisfied by P, then Problem 1 can be solved in polynomial time using linear programming.*

Proof. We can use linear inequalities and constraints of an arc-node formulation to enforce a multicommodity flow $(x_j)_{j \in J}$ where $x_j : E \to \mathbb{R}_+$ assigns each edge a flow value for j. Now introducing variables $\ell_e \geq 0$ with linear constraints

$$c(e, \pi) \sum_{j \in J} d_j(\pi) x_j(e) \leq \ell_e \qquad \forall \pi \in \Pi, \forall e \in E, \tag{3.2}$$

lets ℓ_e equal $\max_{\pi \in \Pi} \left(c(e, \pi) \sum_j \sum_{P_j : e \in P_j} r_j^P d_j(\pi) \right)$ under the objective of minimizing $\sum_{e \in E} \ell_e$. Here, a set of paths P_j as well as path fractions r_j^P arise from a path decomposition of the flow x_j. □

3.2 Special Cases

We consider the following parameters and special cases to analyze the complexity of Problem 1.

- The number of properties $p := |\Pi|$.
- The number of demands k.
- The number of commodities n where the demands D_j can be described as scalar multiples of n commodities $K_1, \ldots, K_n$, with $1 \leq n \leq k$.

ONE-PROPERTY. Each demand has only one property, that is, for each D_j we have $d_j(\pi_j) > 0$ for a property $\pi_j \in \Pi$ and $d_j(\pi) = 0$ for $\pi \in \Pi, \pi \neq \pi_j$. We can think of this case as having k commodities corresponding to the properties.

UNIQUE-PROPERTY. The special case of ONE-PROPERTY where no two demands have the same property. In this case, w.l.o.g. $k = p$.

ONE-TARIFF-TYPE. We thinking of $c(e, \pi)$ as a function $c : e \to \mathbb{R}_+^\pi$. In the ONE-TARIFF-TYPE the cost vectors $c(e, \cdot)$ are then linearly dependent, i. e. $c(e, \pi) = c(e) c_\pi$ for all $\pi \in \Pi, e \in E$ and suitable property costs c_π. W.l.o.g. we can assume $c_\pi \equiv 1$ (Otherwise divide all $d_j(\pi)$ by c_π).

HUBS. The vertex set V allows a hierarchical partition into layers $S, H_1, \ldots, H_\ell, H_{\ell+1}, T$ such that the subgraph of G induced by two different layers is complete bipartite and edges are always directed to the higher layer, S being the lowest and T the highest. We say that this instances has ℓ *consolidation layers*.

We further require each source or sink node to serve only one demand in order to count consolidation layers consistenly. Note, that we can turn multi-demand sources into hubs and outsource the demands to dummy demand nodes connected with some zero cost edge. However this transformation possibly adds one more consolidation layer.

Hardness. For UNIQUE-PROPERTY CONSOLIDATION STEINER SUBGRAPH we can identify properties and demands, i.e. we write $c(e, \pi)$ as $c_j(e)$, and multiply the demands d_j into the cost function c_j. The undirected version of this problem is known in the literature as MULTICOST STEINER SUBGRAPH [JS12].

Problem 2 (MULTICOST STEINER SUBGRAPH)
Given: An undirected graph $G = (V, E)$ and a set of demands $D_1, \ldots, D_k$ with $D_i = (\text{source}_j, \text{sink}_j, c_j)$, where source_j and sink_j are source and sink vertices of the demand, respectively, and $c_j : E \to R_+$ is a cost function.
Goal: Find a list of paths $P_1, \ldots, P_k$ in G, where P_j connects source_j with sink_j and their cost is minimal:

$$\sum_{e \in E} \max_{1 \le j \le k} \sum_{j : e \in \mathcal{P}_j} c_j(e). \tag{3.3}$$

Jordán and Schlotter [JS12] prove that MULTICOST STEINER SUBGRAPH is $\mathcal{NP}$-hard for any constant $k = p \ge 2$, even when each costs are in $\{1, \infty\}$. Moreover, in this setting it is even W[1]-hard for the parameter "cost of the solution". It is also $\mathcal{NP}$-hard on graphs of treewidth 2 (that is, series-parallel graphs).

On the positive side, the problem is fixed-parameter tractable when both k and the treewidth are parameters, using dynamic programming on the tree decomposition. [JS12] also provide a fixed-parameter algorithm for the parameters k and e_s, Here, e_s is the number of edges e where there are j, j' with $c_j(e) \neq c_{j'}(e)$. It is based on dynamic programming using the Dreyfus–Wagner method. The parameter e_s seems to make less sense for our application because it is usually very high, that is, the demands have very diverse properties.

When further the cost functions c_j are identical for all demands, we obtain the well-known STEINER FOREST problem, where we minimize the sum of the weight of edges that are on at least one path. The STEINER FOREST problem is $\mathcal{NP}$-hard even on planar graphs with unit edge costs and on graphs with treewidth three. However, STEINER FOREST is polynomial-time solvable in the case of two terminal pairs, see [Lai+11; JS12].

Most of the hardness results extend to UNIQUE-PROPERTY CONSOLIDATION STEINER SUBGRAPH that is defined on a directed graph, because we can simulate undirected edges by a gadget: replace an edge $\{u, v\}$ by five arcs (u, s_{uv}), (t_{uv}, u) (s_{uv}, t_{uv}), (t_{uv}, v), and (v, s_{uv}). The costs for all arcs are zero except for (s_{uv}, t_{uv}), which gets the cost from (u, v). This reduction preserves all mentioned properties and parameters, with the exception of HUBS. Thus, we get the following result.

Theorem 3.2 UNIQUE-PROPERTY CONSOLIDATION STEINER SUBGRAPH *is $\mathcal{NP}$-hard on planar graphs with unit cost functions $c_j \equiv 1$, or on graphs with treewidth three and identical cost functions c_j.*

The Special Case UNIQE-PROPERTY ONE-TARIFF-TYPE As the directed case is more general than the undirected case, we now turn to the undirected case: We first observe that undirected UNIQE-PROPERTY ONE-TARIFF-TYPE exhibits a notion of cycle freeness.

Lemma 3.3 *W.l.o.g. an optimal solution for the undirected* UNIQUE-PROPERTY ONE-TARIFF-TYPE *can be assumed to be cycle-free, that is, the set of flow carrying edges contains no cycle.*

Proof. First observe that for UNIQE-PROPERTY ONE-TARIFF-TYPE the cost function (3.1) simplifies to $\sum_{e\in E} c(e)\max_{j:e\in P_j} d_j$. Let $P_1,\dots,P_k$ be an optimal solution and let C be a simple cycle in the graph induced by edges in $P_1,\dots,P_k$. For $e\in E$ define $J(e) := \{j : e\in P_j\}$ and $e_{\min} := \min_{e\in C}\max_{j\in J(e)} d_j$. The choice of $e_{\min}$ ensures that joining $J(e)$ with $J(e_{\min})$ on some edge $e\in C$ does not increase the maximum.

For demands j in $J(e_{\min})$ we define alternative paths $\tilde{P}_j$ by pushing flow along C in the direction that cancels the flow of j on $e_{\min}$: Let u_j be the first node and v_j the last node of P_j that belongs to C. We denote with $P_j[u_j,v_j]$ the sub path of P_j from u_j to v_j. W.l.o.g. we can assume that $P_j[u_j,v_j]\setminus C = \emptyset$ because otherwise we alter $P_j[u_j,v_j]$ to use C to connect u with $e_{\min}$ and also to connect $e_{\min}$ with v: By the choice of $e_{\min}$ this does not increase the cost of any edge of C. Further let P_C be $P_j\cap C$ and $\tilde{P}_C$ be $C\setminus P_C$. Then we set $\tilde{P}_j = P_j[\text{source}_j,u_j]+\tilde{P}_C+P_j[v,\text{sink}_j]$ and see that $\tilde{P}_j$ avoids $e_{\min}$. By the previous observation no edge of C is used more than once by j. If we replace P_j with $\tilde{P}_j$, then again $c(e)\max_{j\in J(e)}$ does not increase on edges e affected by this rerouting.

After rerouting all demands in $J(e_{\min})$, the edge $e_{\min}$ is no longer used we have canceled one cycle. Also no new cycles are introduced as our rerouting is restricted to edges of C. This method can be repeatedly applied to cancel simple cycles to turn an optimal solution into a cycle free solution of no greater costs. □

We now consider the undirected UNIQUE-PROPERTY ONE-TARIFF-TYPE case with $k = p = 2$. Note that the hardness from STEINER FOREST does not apply here, since it requires p to be unbounded.

Problem 3 (UNIQUE-PROPERTY ONE-TARIFF-TYPE, $k = p = 2$)
Given: A undirected graph $G = (V,E)$ and two demands $(\text{source}_1,\text{sink}_1,d_1)$ and $(\text{source}_2,\text{sink}_2,d_2)$ and a cost function $c : E\to R_+$.
Goal: Find two paths P_1 and P_2, where P_j connects source_j with sink_j and their cost is minimal:

$$\sum_{e\in P_1} d_1 c(e) + \sum_{e\in P_2} d_2 c(e) - \sum_{e\in P_1\cap P_2} \min\{d_1,d_2\}c(e). \tag{3.4}$$

Lemma 3.4 *The* UNIQUE-PROPERTY ONE-TARIFF-TYPE *case with $k = p = 2$ can be solved in polynomial time.*

Proof. The proof is similar to ideas by [Lai+11]. By Lemma 3.3, no cycles are required in the solution. There is an optimum that is either a forest consisting of two shortest paths, or a tree connecting all four terminals. Such a tree has at most two vertices of degree 3 and we can enumerate all possible pairs for these two vertices in $O(|V|^2)$. The corresponding tree cost can be obtained computing a constant number of shortest paths in each case. □

Unfortunately, the more general case, ONE-PROPERTY ONE-TARIFF-TYPE where we allow a property to show up in more than one demand, is already hard.

Lemma 3.5 *The* ONE-PROPERTY ONE-TARIFF-TYPE *case with $p = 2$ is $\mathcal{NP}$-hard.*

Proof. We reduce from Problem 4:

> **Problem 4 (PARTITION)**
> **Given**: $N+1$ integer numbers $a_1, \ldots, a_N, K$ with $\sum_j a_j = 2K$
> **Goal**: Decide whether there is $S \subseteq [N]$ such that $\sum_{j \in S} = K$

We construct a ONE-PROPERTY ONE-TARIFF-TYPE instance and choose the property cost vector for the one tariff type to be $(1,1)$. Thus, the cost incurred on any edge e can be written as $c(e) \max_{\pi \in \Pi} \sum_{j: e \in P_j} d_j(\pi)$. We now build a graph with three nodes u, v, w, three edges $\{u,v\}$, $\{u,w\}$, $\{v,w\}$ and with edge cost $c(\{u,v\}) := 3$, $c(\{u,w\}) := 2$, $c(\{v,w\}) := 2$. Furthermore, for each number a_j we introduce a demand $D_j := (u, v, (a_j, 0))$ and one additional demand $D_{N+1} := (u, w, (0, K))$. We now prove the following: The given PARTITION is a "yes" instances if and only if the constructed graph admits a solution of cost $7K$.

"$\Rightarrow$" Let $J \subset [N]$ be a partition such that $\sum_{j \in J} a_j = K$, then routing demand D_K along edge $\{u,w\}$, demands D_j, $j \in J$ along the path $\{u,w,v\}$ and finally demands D_j, $j \in [N] \setminus J$ along nodes $\{u,v\}$ has the required cost of $3K + 2K + 2K = 7K$

"$\Leftarrow$" If for every subset $S \subset [N]$ it holds that $c(S) \neq K$, than we show that any routing in the graph has costs strictly greater than $7K$. Let $J \subseteq [N]$ be the set of demands that are routed along $\{u,w,v\}$ and $\sum_{j \in J} a_j =: C$. Also let $c_J(e)$ be the cost incurred on edge e for the specified routing. There are 4 cases:

Case 1 D_{N+1} is routed along $\{u,v,w\}$ and $C > K$: We have $c_J(\{u,v\}) = 3K$, $c_J(\{u,w\}) = 2C$, $c_J(\{v,w\}) = 2C$, and altogether a cost value of $3K + 4C > 7K$.

Case 2 D_{N+1} is routed along $\{u,v,w\}$ and $C < K$: We have $c_J(\{u,v\}) = 3(2K - C)$, $c_J(\{u,w\}) = 2C$, $c_J(\{v,w\}) = 2K$, and altogether a cost value of $7K + (K - C) > 7K$.

Case 3 D_{N+1} is routed directly along $\{u,w\}$ and $C > K$: We have $c_J(\{u,v\}) = 3(2K - C)$, $c_J(\{u,w\}) = 2C$, $c_J(\{v,w\}) = 2C$, and altogether a cost value of $6K + C > 7K$.

Case 4 D_{N+1} is routed directly along $\{u,w\}$ and $C < K$: We have $c_J(\{u,v\}) = 3(2K - C)$, $c_J(\{u,w\}) = 2K$, $c_J(\{v,w\}) = 2C$, and altogether a cost value of $7K + (K - C) > 7K$.

□

Lemma 3.6 *The* ONE-PROPERTY ONE-TARIFF-TYPE *case with $p = 2$ for directed graphs is strongly $\mathcal{NP}$-hard.*

Proof. Reduction from Problem 5:

> **Problem 5 (3-PARTITION)**
> **Given**: A set of integer numbers $a_1, \ldots, a_{3m}$ and an integer B with $\sum_{i \in [3m]} a_i = mB$ and $m/4 < a_i < m/2$
> **Goal**: Decide whether there are sets $S_j \subset [3m]$, $j = 1, \ldots, m$ such that $\sum_{i \in S_j} a_i = B$ for all j

We construct a ONE-PROPERTY ONE-TARIFF-TYPE instance I and choose the property cost vector for the one tariff type to be $(1,1)$. The graph has a special node s, one common sink node t and an intermediate layer of m nodes u_j, j in$[m]$. For each number a_i we introduce a demand $(s,t,(a_i,0))$ and additionally we introduce m demands $(u_j,t,(0,B))$ for j in $[m]$. Finally we have edges (u_j,t) of cost $c((u_j,v_j))=1$ and edges (s,u_j) of cost zero, for each j in $[m]$.

The cost of any solution to instance I is by construction lower bounded by mB because the demands $(u_j,t,(0,B))$ can only be routed along edges (u_j,t). We show that 3-PARTITION is a yes instance if and only if I admits a solution of cost equal to mB .

"⇒" Let $S_j, j \in [m]$ be a solution of 3-PARTITION. Then a joint routing of the three demands $(s,t,(a_i,0))$ for each $i \in S_j$ along the path $\{s,u_j,t\}$ as cost B. Realizing such a routing for all $j \in [m]$ implies overall costs of mB.

"⇐" Let P_i be the path of demand $(s,d,(a_i,0))$ in a solution of overall cost mB. The solution cost achieves the lower bound implied by the $(u_j,t,(0,B))$ demands. Thus none of the edges (u_j,t) can have routing costs greater than m. From $\sum_{i\in[3m]} a_i = mB$ it follows that the demands $(s,d,(a_i,0))$ must be distributed on the edges (u_j,t) such that the maximum property cost of m is attained for both properties on each edge (u_j,t) and thus the sets $S_j := \{i : (u_j,t) \in P_i\}$ are a feasible solution to the 3-PARTITION instance.

□

3.3 Dynamic Programming for Uniform Consolidation Steiner Subgraph

We now turn to the HUBS Special Case and address it with dynamic programming. As our dynamic program does not specifically use properties of the cost funcions, we formulate it for Problem 6 that generalizes the ONE-TARIFF-TYPE in that the cost of the tariff type is a subadditive set function given by an oracle. Subadditivity can be understood as a notion for consolidation effects or economies of scale for transportation costs ensuring that the costs cannot decrease, if the set of transported demands increases.

Problem 6 (UNIFORM CONSOLIDATION STEINER SUBGRAPH)

Given: A directed graph $G=(V,E)$; A list of demands $J=\{D_1,\dots,D_k\}$ with $D_j = (\text{source}_j, \text{sink}_j)$; cost functions $c_e : 2^J \to \mathbb{R}_+$ of the form $c_e(S) := l_e f(S)$ with $S \subseteq J$ and $f : 2^J \to \mathbb{R}_+$ subadditive i.e.

$$f(A \cup B) \leq f(A) + f(B) \; \forall A, B \subseteq J, \tag{3.5}$$

and $l_e : E \to \mathbb{R}_+$ an edge length function.

Goal: Find a list of paths $P_1,\dots,P_k$ in G, where P_j connects source_j with sink_j and their costs are minimal:

$$\sum_{e\in E} c_e\big(\{j \in J : e \in P_j\}\big). \tag{3.6}$$

First note, that cycle freeness, as ensured in Lemma 3.3 for the special case of undirected UNIQUE-PROPERTY ONE-TARIFF-TYPE, does not apply to the directed case: For example satisfying demands that are suitable positioned on a a cycle graph can always require the support graph of used paths to be the cycle graph. Instead we propose the notion of forest solutions.

Definition 3.1 (Forest Solutions) *A solution $P_1, \dots, P_k$ together with a partition $\mathcal{T} = (T_1, \dots, T_r)$ of J into $r \leq |J|$ subsets T_i is called a* forest solution*, if the undirected version of the graph built by the support of the paths P_j, $j \in T_i$ is a tree for each i. We also define the* forest cost *by*

$$\tilde{c}(\mathcal{T}) := \sum_{T \in \mathcal{T}} \sum_{e \in E} c_e\big(\{j \in T : e \in P_j\}\big), \tag{3.7}$$

that is, consolidation effects only apply to demands from same subsets T_i.

We think of forest costs as the costs of routing the demands of each T in the partition within a separate round. This way we overestimate the original costs on edges that are used in several rounds. In the following, we abuse notation by referring to some demand set T with the desired property as a tree.

Hubs Special Case with Two Consolidation Layers

With this restriction each source-sink path in some solution can have at most two edges, that carry more than one demand, i.e. edges between H_1 and H_2 or between H_2 or H_3.

Lemma 3.7 *For the* HUBS *special case with two consolidation layers one can restrict to forest solutions.*

Proof. Consider some optimal solution $P_1, \dots, P_k$. The first set of trees $(T_v)_{v \in H_2}$ to be used in $\mathcal{T}$ arises from nodes in H_2. Define $J(e) := \{j \in J : e \in P_j\}$ and $J(v) := \bigcup_{e \in \delta^-(v)} J(e) \bigcup_{e \in \delta^+(v)} J(e)$ and set $T_v := J(v), v \in H_2$. Observe that for two distinct nodes $u, v \in H_2$ the two edge sets $\bigcup_{j \in J(u)} P_j$ and $\bigcup_{j \in J(v)} P_j$ are disjoint and their their undirected versions build trees.

Now, let $J^r := J \setminus \bigcup_{v \in H_2} J(v)$ be the remaining demand set. Then for each $v \in H_1$, the set of edges used of paths $P_j, j \in J(v) \cap J^r$ is disjoint with each of edge sets considered before and with the corresponding edge sets for other nodes $u \in H_1$ thus we also use Trees $T_v := J(v) \cap J^r, v \in H_1$. Finally, all remaining demands in J^r must be routed along paths not containing any node in H_1 or H_2 which means, due to the special layered structure, that none of the edges in these paths carries more then one demand. Thus, the sets $J(v)_{v \in H_2}$, $J(v)_{v \in J(v) \cap J^r}$ and all remaining demand singletons form the partition for a forest solution. □

Uniform Consolidation Steiner Subgraph with ℓ Consolidation Layers

We have just seent that restricting to forest solutions is feasible for the HUBS special case with two consolidation layers. Example 3.1 shows that this is wrong for HUBS Special Case with three consolidation layers, even in a single sink version.

Example 3.1 (Forest solutions are not sufficient) *Consider the graph depicted in Fiqure 3.1 with three consolidation layers and four demands $D_1 = (s, t, (1, 0)), D_2 = (s, t, (0, 1)), D_3 =$*

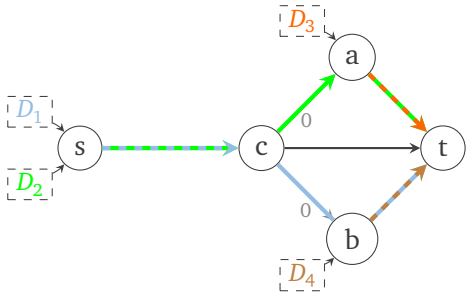


Figure 3.1: Network for Example 3.1

$(a,t,(0,1)), D_4 = (b,t,(1,0))$. *Edge costs $c(e)$ equal 1 except for edges connecting sources or edges labeled with 0 that have zero costs. We have depicted an optimum routing where consolidation effects on dotted edges $(s,c),(a,t),(b,t)$ allow total costs of 3. Among optimum partitions for forest solutions we find $\mathcal{T} = \{\{1,2\},\{3\},\{4\}\}$ and $\mathcal{T}' = \{\{1\},\{2\},\{3,4\}\}$, each with forest objective of 4 because consolidation effects can be realized either on edges (s,c) and (c,t) for $\mathcal{T}$, or on edges (a,t) and (b,t) for $\mathcal{T}'$. While $\mathcal{T}'$ leads to the optimium for the true objective, $\mathcal{T}$ does not.*

This example also shows, that forest solutions can be off by a factor of $4/3$ of the optimum. However, proving some upper bound remains open. In the following we aim at providing a dynamic program that finds at least forest solution that have optimal forest costs for instances of Uniform Consolidation Steiner Subgraph with bounded number of consolidation layers.

Definition 3.2 (Join, Split) *Let $j(v,\ell,S)$ be the minimum cost for routing demands $S \subseteq J$ from their sources to v along an in-tree rooted at v of height ℓ and let $s(v,\ell,S)$ be the minimum cost of routing demands S from v to their sources along an out-tree rooted at v of height ℓ (assume infinite costs if such a routing is impossible, e. g. for trees of heigth 0).*

Lemma 3.8 *$j(v,\ell,S)$ fulfills recursion formulas:*

$$j(v,\ell,S) = \min_{\mathcal{T}\subseteq 2^S : \dot{\bigcup}_{B\in\mathcal{T}} B = S} \sum_{B\in\mathcal{T}} \tilde{j}(v,\ell,B) \tag{3.8}$$

$$j(v,0,S) = \tilde{j}(v,0,S) := \begin{cases} 0 & \text{if } S = \{j\} \text{ and } v \text{ is source of demand } j \\ \infty & \text{otherwise} \end{cases} \tag{3.9}$$

$$\text{and for } \ell \geq 1 \quad \tilde{j}(v,\ell,S) := \min_{(u,v)\in A} j(u,\ell-1,S) + c_{(u,v)}(S) \tag{3.10}$$

Analog recursion formulas can be derived for $s(v,\ell,S)$.

Proof. We proof Equation (3.8) by induction on ℓ. For $\ell = 1$, $j(v,1,S)$ is the cost along a unique in-tree rooted at v of height one, that is the collection of (source_j, v) edges for $j \in S$. This routing cost is attained in Equation (3.8) by a partition $\mathcal{T}$ of S into singletons and the respective values $\tilde{j}(v,1,\cdot)$. Note, that for some non-singleton $B \subseteq J$, $\tilde{j}(v,1,B)$ equals infinity since we required all sources to serve one demand only. So all other partitions $\mathcal{T}$ involving non-singleton sets in Equation (3.8) will have $\sum_{B\in\mathcal{T}} \tilde{j}(v,1,B)$ equal to infinity.

Now for $\ell > 1$, consider some optimal routing of demands S to v on an in-tree T^* of height ℓ and let $c(T^*)$ denote the sum of edge costs in this tree. Let furthermore F be the edges in $\delta^-(v)$ that carry some demands. Clearly, for each edge $(u,v) \in F$, the demands $J((u,v))$ are routed to u on an in-tree $T_{(u,v)}$ rooted at u of height $\ell - 1$, so if $c(T_{(u,v)})$ denotes the routing cost of this subtree, then clearly $c(T^*) = \sum_{(u,v)\in F} c(T_{(u,v)}) + c_{(u,v)}(J((u,v)))$. By induction, $c(T_{(u,v)}) = j(u, \ell-1, J((u,v)))$ for each $(u,v) \in F$ and since $(J((u,v)))_{(u,v)\in F}$ is a partition of S, $' \geq'$ follows in Equation (3.8). To see $' \leq'$ consider some partition $\mathcal{T}'$ of S and assume for all $B \in \mathcal{T}'$ that $\tilde{j}(v,\ell,B)$ is finite. Otherwise this partition yields infinity for the sum in (3.8). Let (u_B, v) be the edge where equation (3.10) attains its minimum for $\tilde{j}(v,\ell,B)$. Then $\sum_{B\in\mathcal{T}'} \tilde{j}(v,\ell,B)$ corresponds to the cost for routing demands B to u_b along some minimum in-tree of height $\ell - 1$ for each $B \in \mathcal{T}'$ (by induction), and then for routing demands B along (u_b, v) to v. Thus the cost corresponds to a routing along an in-tree of height ℓ and is lower bounded by the optimal in-tree cost. □

Lemma 3.9 *Given $\tilde{j}(v,\ell,S)$ for fixed ℓ, fixed $v \in V$, and for every $S \subseteq K$, then one can compute Equation (3.8) in $O(|2^K|^2)$.*

Proof. One can show by induction on $|S|$ that

$$j(v,\ell,S) = \min\left\{\tilde{j}(v,\ell,S),\ \min_{\emptyset\subset A\subset S} j(v,\ell,A) + j(v,\ell,S\setminus A)\right\}. \tag{3.11}$$

So we can store the $j(v,\ell,S)$-values for all S in a dynamic programming table of size 2^J where the values are ordered with respect to increasing cardinality of S. In order to compute $j(v,\ell,S)$, it suffices to iterate over the entries $j(v,\ell,A)$ for all $A \subset S$ of smaller cardinality, which results in the specified running time. □

Theorem 3.10 *Algorithm 3.1 finds forest solutions $\mathcal{T}$ where for each $B \in \mathcal{T}$ the height of in-subtrees and out-subtrees of B is bounded by ℓ with a running time in $O(\ell(|E||2^J| + |V||2^J|^2) + |V||2^J|^2))$.*

Proof. The Lines 2 to 14 implement the recursion formulas (3.8) to (3.10). According to Lemma 3.8 the values $\mathsf{tree}(S)$, which are updated in Line 14, represent the minimum costs of routing demands S from their sources along an in-tree, which is rooted at some node $v \in V$, and then to their sinks along an out-tree routed at v. Both trees have their height bounded by ℓ. In Line 15 we optimize a partition of j that combines the tree routings of subsets B with cost $\mathsf{tree}(B)$ to a routing for J. The result is an optimal forest solution.

However, for the running time the step in Line 15, we can use the arguments of Lemma 3.9 and show by induction on $|S|$ that $\mathsf{sol}(S)$ can be computed recursively by iterating over all smaller subsets of S:

$$\mathsf{sol}(S) = \min\left\{\mathsf{tree}(S),\ \min_{A\subset S}\{\mathsf{sol}(A) + \mathsf{sol}(S\setminus A)\}\right\}$$

A straight forward dynamic program then requires $O(|2^J|^2)$ operations to compute $\mathsf{sol}(J)$ in Line 15. This explains the last additive term of the running time whereas the loops in line 2 to 14 contribute the first two addends. □

Algorithm 3.1: Dynamic Programming

Input: Instance of UNIFORM CONSOLIDATION STEINER SUBGRAPH
Output: A Forest-Solution with optimal forest cost

assume all $j, s, \tilde{j}, \tilde{s}, \mathsf{tree}(S)$ values initialized to infinity or according to (3.9);
for $m = 1, \dots, \ell$ **do**
  for $(u,v) \in E$ **do**
    for $S \subseteq J$ **do**
      $\tilde{j}(v,m,S) = \min\{j(v,m-1,S),\ j(u,m-1,S) + c_{(u,v)}(S)\}$;
      $\tilde{s}(u,m,S) = \min\{s(u,m-1,S),\ s(v,m-1,S) + c_{(u,v)}(S)\}$;
  for $v \in V$ **do**
    for $S \subseteq J$, *in order of increasing cardinality* **do**
      $j(v,\ell,S) = \tilde{j}(v,\ell,S)$; `// see Lemma 3.9`
      for $A \subset S$ *with* $|A| = 1, \dots, \lfloor |S|/2 \rfloor$ **do**
        $j(v,\ell,S) = \min\{j(v,\ell,S), j(v,\ell,A) + j(v,\ell,S \setminus A)\}$;
      update $s(v,\ell,S)$ analogously;
      if $m = \ell$ **then**
        $\mathsf{tree}(S) = \min\{\mathsf{tree}(S), j(v,\ell,S) + s(v,\ell,S)\}$;
return $\mathsf{sol}(J) := \min_{\mathcal{T} \subseteq 2^J : \dot{\bigcup}_{B \in \mathcal{T}} B = J} \sum_{B \in \mathcal{T}} \mathsf{tree}(B)$;

Chapter 4

Robust Strategic Route Planning

4.1 Introduction

Logistics is a highly cost sensitive industry. In this chapter we pursue optimization of strategic planning from the perspective of an inbound logistics' customer. This means a company—in our real-world example an automotive supplier—wants to route different goods from different suppliers to its factories at minimal cost given the transportation tariffs offered on the market by different freight companies. These routes are chosen and fixed (up to very rare and small changes) in the strategic planning phase, i.e., usually a year ahead. The routes can be consolidated with each other at intermediate hubs. In most inbound logistic networks hubs are only used for consolidation of routes, not for storage. This is the only reason why we restrict to inbound logistics here.

As even small changes in cost are highly relevant, the optimization model should contain a accurate cost functions. Moreover, the strategic planning takes place before important data, in particular the demands, are known precisely, and the strategic plan has to work well for many weeks, each with a different demand not known before. We show in this chapter that under such conditions using optimization under uncertainty is equally important as an accurate cost model.

The concept for optimization under uncertainty chosen here is robust optimization, i.e., to find a routing that minimizes the cost of its restricted worst-case. The application of the industry partner which provided for the test cases, the 4flow AG [4fl17], yields two reasons for choosing robust optimization. First, the company is a logistics consultancy company managing logistic networks for its customers, in this case, the automotive supplier. Hence, the company is interested in a cost guarantee for any reasonable demand scenario, i.e., to minimize the restricted worst-case cost. Second, the strategic choice of routes will be operated during several weeks, and should not cause excessive costs in any of those weeks. From a management perspective it is not sufficient for a solution to be good on average.

In this chapter we present a robust optimization model for this problem. We show that already the adversary problem is $\mathcal{NP}$-hard. Still, we develop a mixed integer program approximating the robust model. The approach is solvable for large-scale real-world instances. In the computational study we evaluated the approximative solution with its exact robust (and nominal) cost.

Technically the underlying problem is a multicommodity flow, where each commodity is

a triplet of supplier, factory, and good, which we call a demand. The flow has no practically relevant capacity restriction. The difficulty of the problem stems from the cost function. An accurate model of the market tariffs should contain the following two features: First, there are discrete economies of scale. The tariff levels can be understood in analogy to physical containers: the price of a container is constant no matter which percentage is filled. Hence the cost function has discrete jump points. Further, higher tariff levels correspond to larger containers that come at a lower cost per size. Hence there are economies of scale. The second feature of the cost function makes it advantageous to consolidate routes of heavy and lightweight goods. To assess the capacity of a container one has to evaluate two (or more) properties of the shipment, e.g., the maximal weight and the maximal volume. The size required for a shipment is the maximum required size over all properties.

Fluctuating demands are the most influential parameters among the uncertain data. We model their fluctuation by scenario sets described by an interval for the size of each demand derived from historic data. Further, we restrict the number of demands affected by deviation from the nominal value in a scenario. This amounts to fixing the number of demands deviating to their maximal size in the worst case, while all other demands stay at their nominal value. Note, increasing the demand, e.g., for a heavy, small good will not have a strong effect on the cost of a link where the container size is determined rather by volume than by weight, i.e., where there is a lot of slack in the weight restriction. Thus, finding the worst-case scenario, i.e., the adversary problem needs to take the maximization over the properties into account. This is what makes the adversary problem difficult.

The key ingredients to attain a robust model solvable for large real-world networks is a linear relaxation of the adversary problem and a partial linearization of the economies of scale. With these relaxations we can formulate the robust problem as a mixed integer program that is in some respects similar to a facility location problem, and like facility location IPs tends to have optimal solutions very close to those of the LP. We assess the quality of the solution by solving the adversary problem exactly and using the exact cost function.

Chapter outline We start by a more detailed description of the application motivating this work in Section 4.2, followed by a review of related work. In Section 4.4 we formulate a mathematical model for the logistic routing problem without uncertainties in the input data. In Section 4.5 we extend this to a robust model, and show that the included adversary problem is NP-hard. As these exact models are far from being solvable in particular for the size of real-world instances we develop careful simplifications in Section 4.6 to obtain a solvable robust model accurate enough to yield practically useful solutions. In Section 4.7 we explain in detail the computational methods used to actually solve the test cases. The result of these computations are discussed in Section 4.8. In Section 4.9 we summarize the heuristic methods with which we have tested to extend the size of solvable instances even further and present the results of these computations. Finally, in Section 4.10 we test our methods under real demand date by splitting the historical data into a training and a testing set.

4.2 Problem Statement: A Customers' View

In this section we describe the applications our model is based on.

4.2.1 Inbound logistic networks

The running example in this chapter is the inbound logistics network of an automobile manufacturer. We orient our modeling by this example to guarantee practical relevance of our method. Still, the model is fairly general for routing in hub networks. Although this instance is large, we study even larger networks to explore the limits of the approach.

The manufacturer runs 23 assembly plants in Europe, which receive about 10^4 different types of parts on a more or less regular basis during the planning horizon. These parts are provided by 400 suppliers. They are transported directly or via intermediate hub nodes that allow for consolidation of the shipments.

Each transportation task is typically outsourced to freight forwarders. The freight forwarders offer *Transport Relations* between hubs, plants, and suppliers, i.e., we have a directed, multicommodity network with plants as *sinks*, suppliers as *sources*, hubs as additional nodes, and Transport Relations as *edges*.

For the set of commodities we consider all articles transported through the network or classes of articles, when the differences of the articles are negligible. Formally, we speak of a *demand* meaning a triplet of source and sink node together with an article (or article class). By J we denote the set of all demands. For this set the vector of *demand values* $d \in \mathbb{R}_+^J$ gives the amount of demand units to be transported for each demand. Further, as the transportation cost depends on certain properties of the articles of some set Π, we need for each demand α_j the *property vector* of demand j. Its components $\alpha_{j\pi} \in \mathbb{R}_+$ denote the per unit capacity for property $\pi \in \Pi$ needed for the articles of demand j. For example, if property π is the volume, then demand j consumes $d_j \alpha_{j\pi}$ space in a truck.

For practical reasons, a demand should not be split during the routing. Also, capacity restrictions on the transport relations in practice are too high to become relevant.

We want to design the routing in the network to minimize transportation costs. Thus, we face an unsplittable, uncapacitated, multicommodity flow problem with a complex cost function which we detail next.

4.2.2 Transportation cost

Benchmark tariffs Real world transportation tariffs for Transport Relation may be very complex and diverse throughout the freight market. In the strategic planning phase, which we consider here, it is nearly impossible to tender all the possible transports beforehand. Therefore, we use *benchmark tariffs* for different full truck load (FTL) and less than truck load (LTL) transports, that are used in the industry as well and yield a sufficiently accurate tariff model. In particular, these tariffs cover two major effects when commodity flows share an edge in the network:

- *consolidation effects*: loading commodities of complementing properties together may save transportation cost
- *economies of scale*: marginal shipping costs decrease when many demands are combined to large shipments

For a fixed edge e of the network the (benchmark) tariff offers a set of tariff levels $K(e)$. Each such tariff level can be thought of as a type of *container* with a fixed price per container g_k. Each container type $k \in K(e)$ has capacities $\beta_{\pi k}$ for all properties π in Π. Several containers

of a certain type can be used for one shipment. Yet, once this *multiplicity* f_k becomes too large, it is cheaper to use a larger container type.

Tariff selection In theory, one shipment on one edge could be distributed onto several container types with different multiplicities to minimize cost. This gives rise to the so-called *Tariff Selection* (TS) which is obviously $\mathcal{NP}$-hard. In [KMR12] a $\mathcal{NP}$-hardness proof is given even when considering only one property in the model. Also very fast heuristics for the general case that allow the embedding of the subproblem into a local search heuristic are proposed. In practice, using only one tariff and container type on each edge is common, i.e. $|\{k \mid f_k > 0\}| = 1$. As it is industry standard we adopt this restriction throughout this chapter. This *Single Tariff Selection* (STS) trivially becomes polynomial solvable by enumerating over all tariff levels.

Handling and storage cost Storage cost at suppliers is not part of the consideration in this chapter. Moreover, storage at hub nodes is typically avoided in inbound logistics networks. Hubs are usually not run by the company but rented. Rental contracts for hubs today usually do not include opening cost. Instead costs for renting hubs depends on the quantities transferred through the hub. We can assume per unit handling cost rates for each article. Thereby, the handling costs are structurally similar to transportation cost when we conceive of a hub as a directed arc.

4.2.3 Hierarchical planning and uncertainty

Strategic planning Logistic planning typically proceeds in three planning phases: the strategic, tactical and operational planning phases. This chapter deals with the *strategic planning* in which decisions with long term effects are taken, i.e., they affect the next years. In the strategic phase only rough forecasts based on historical data for the occurring delivery requests are available. Also, not all details have to be fixed in the strategic phase, but are left to the tactical planning.

In our case, the strategic planning has to decide the routing, i.e., to find a shipment path for each demand in the network. The hubs and the connections along this path are fixed in the strategic planning. Framework contracts with external freight forwarders can be negotiated based on the resulting expected shipping volumes on each transport relation. Changing the routing decisions or even introducing new hubs in later phases is mostly avoided due to the managerial overhead.

The strategic decisions form the basis for the tactical and operational planning phase which have more or less exact information on the demands. The framework contracts allow to adjust quantities of shipment along the fixed paths in the tactical phase, e.g., on a weekly basis. It is important to note, that the weekly adjustment to the transported quantities also allows to optimally adjust the choice of tariff level.

Optimization under uncertainty During strategic planning future weekly demand values are unknown. The planning has to work with forecasts from historical data. The strategic decisions for the routing remain valid for the whole planning horizon. But, the actual costs of these long term decisions are uncertain because they depend on the short-term adjustments to the different the realizations of demand values over the planning horizon.

Therefore, we face a two-stage optimization problem: In the first stage, the strategic planning, we have vague information on the demands, and have to fix a routing. In the second stage, the tactical planning, we have fairly accurate information on the demands and can adjust the Single Tariff Selection.

A second uncertain parameter are the exact tariffs. We do not consider optimization under uncertainty about tariffs in this chapter, as we have seen in a separate study with similar methods, that reasonable changes in tariffs do not have a strong influence on the routing, i.e., the first stage decision.

Finally, the strategic planning is done by a fourth party, a logistics consultancy. For the contract between the consultancy and the autopart manufacturer a tangible guarantee of quality is desirable. This can be achieved by a robust solution guaranteeing a maximal bound on the cost as long as the demands stay within an intelligibly and implicitly given set of scenarios. Robust optimization aims to find a routing with the cheapest of such guarantees. Further, this guarantee will not be exceed in at any time during the planning horizon as long as the demand stays in the defined range. This increases the financial stability for the customer.

4.3 Literature Review

In this section we compile related work for similar logistic problems of routing and hub location with concave costs, and for robust optimization.

4.3.1 Concave Multi-commodity Flow and Network Design

We consider the network flow of multiple commodities a central aspect of our problem. Economies of scale, when restricted to one property only, can be expressed using concave flow dependent cost functions on the network edges. A seminal paper by [GP90] discusses complexity issues, applications and solution techniques for a variant where the assignment of the demands from sources to sinks can be arbitrarily resolved. Also the application in [Kli90] that uses piecewise linear concave cost inspired our modeling as well the observation therein, that the linear relaxation of the presented formulation tends to be strong when applied to problem instances from practice.

The case of concave piecewise linear cost functions can naturally be modelled as fixed charge network flow. One classical and one more recent reference is given by [MW84] and [Cra00] respectively. Recent progress over the state of the art of algorithms and solution techniques has been presented in [HNS10].

In our application we focus on a customer's perspective of strategic transportation planning task. However the converse perspective is that taken for example by LTL freight transportation carriers, where a customer appears as a client. The aim is the design of load plans for the occurring demands of many clients. [Cra+14] and [Ere+12] present such models for the tactical level where temporal and structural side constraints for the design of load plans need to be respected. In both, a time expansion of the network is used and transportation cost is modelled by integer trailer multiplicities. A prior contribution concerning tactical transportation planning for the customer's perspective has been subject of [Har+16], that already exhibits a very detailed modeling of transportation and storage costs.

4.3.2 Hub Location

Side constraints, like a maximum number of allowed hub nodes in a solution relate to hub location problems. For a survey on hub location models see [AK08] or [CEK02]. In these more abstract models, economies of scale that may be realized on inter hub edges only, are often modelled with a uniform discount factor $\alpha < 1$ that is applied to the transportation cost.

Also it is commonly assumed that every demand must be connected to one opened hub, which generates two effects of opening a hub: It can avoid costly detours for some demands and it can enable cost saving detours when demand-paths include a strongly discounted edge. Our problem is substantially different in that we assume a direct connection is always allowed for a demand, leaving only the latter effect relevant. Though our models easily allows such side constraints, their presence highly affects the performance of our algorithms and opens another chapter that we do not addressed in this work. However, the authors in [CO12] identify "a promising opportunity in transportation to better link hub location to network design in general and to tactical transportation service network design in particular".

4.3.3 Robust Network Design

The literature presented so far is concerned with deterministic models, where the input data is assumed to be known. Important applications however face uncertain data. The concept of robust optimization tries to overcome this deficiency by introducing uncertainty sets for these data. [BS03] show how the use of budgeted uncertainty sets may allow for tractable robust counterparts of tractable problems, e.g. if data uncertainty appears in coefficients of the objective function only.

In [Ben+04], a two stage robust model for linear programming is proposed, that is, a set of second stage variables can adapt to the realizing uncertain demand values in contrast to first stage variables remaining fixed. The authors show how a restriction of these adjustments to affine functions of the uncertain data may allow tractable optimization problems. The approach presented in this chapter uses techniques presented from both, [BS03] and [Ben+04].

We are aware of several successful applications of robust optimization to network design for telecommunication networks. Here the task is to install link capacities or facilities at minimum cost to allow a multicommodity routing for any realization of demand values in predefined uncertainty sets. The works differ in the type of uncertainty sets and in the models for adjusting the commodity flow, once their demand values realized.

The work of [PR13] focuses on the model of adjustments and investigates static routing, in which only the flow value adjusts to the realized demand value, dynamic routing, in which the flow pattern can change arbitrarily, and affine routing, in which the flow pattern is expressed as an affine linear function of the realizing demand. In their model the capacities can be installed fractionally incurring linear edge dependent cost.

The model in [AYP11] uses integer capacity installations and allows for two different facilities on each edge, thereby modeling some economies of scale. The branch-and-cut framework proposed uses polyhedral insights for the specific uncertainty sets considered and turns out to be very efficient.

Both contributions are success stories for robust optimization in which the cost for capacity

installation can be reduced by explicitly taking into account uncertainty of the data. In our applications, however the role of first and second stage decisions is interchanged: our routing pattern should remain fixed, while the edge capacities, or in our case the tariff selection, are adjusted to the realizing demand of the commodities. In the above terminology our model can be phrased as static routing with adjustable capacities, the capacity cost however depending on multiple properties and reflecting economies of scale.

4.3.4 Robust Network Flows

A different setting arises if not demand values but edge costs are uncertain. In the limit case we can think of this as edge failures, i.e. edges become too expensive to be used. The resulting flow problems are known as robust network flows in the literature [ACN01]. A basic version asks for a robust counterpart to the classic maximum flow problem: Given a network and an integer k, the task is to find a flow such that its value is maximum after k edges fail according to the flow specific worst case. $\mathcal{NP}$-hardness for this problem is shown by [DM17] for arbitrary k as part of the problem input whereas for $k = 1$ the problem is solvable in polynomial time.

[BNS13] investigate a tractable variant, where instead of a global parameter k, node specific parameters limit the number of edge failures on a node's incoming edges. They also introduce *adaptive* network flows, where the flow can be adapted after the edge failures are revealed as long as new flow values on individual edges do not exceed original flow values. The authors prove $\mathcal{NP}$-hardness for this problem and the existence of an equilibrium when it is viewed as a two-person zero-sum game.

Another tractable model for robust network flows has been introduced by [Mat+17] for path-based flows. In this model a maxium flow is sought that is immunized against targeted attacks of an adversary that attacks edges in the network to steal flow from paths using such edges. The adversary has a budget for stealing flow where the budget costs are specific to the edges. Basic variants of the problem are solvable with a parametric LP whereas for a more general variant a strong inapproximability result is shown.

[MMO17] consider a variant of *reroutable* network flows where on the one hand new flow values can exceed the original flow values but on the other hand only the part of the flow that is affected by the arc failure can be rerouted. They distinguish strict reroutable flows, which are solvable by a compact linear program, from reroutable flows which are $\mathcal{NP}$-hard even when all edge capacities are in {1,2}.

4.4 Deterministic Model

As a deterministic problem, i.e., assuming exact knowledge of the demand values $d \in \mathbb{R}_+^J$, the strategic planning faces an unsplittable, uncapacitated, multicommodity flow problem (MCF). Typical models for MCF problems include arc-node as well as path formulations. In this work we only investigate a path formulation that we solve with a column generation technique.

A computationally competitive path formulation typically necessitates algorithms, that iteratively generate new path variables for each commodity, such as branch-and-price algorithms. Developing a full-fledged branch-and-price algorithm is a complex task that is beyond the scope of this work, whose focus relates to efficiently handling complex tariff structures and data uncertainty.

Also, it is beyond the scope of this work to establish whether a path or an arc-node formulation is more suitable for the pertaining problem. We prefer a path over an arc-node formulation for the following reasons: Unsplittable flows can be easily achieved in a path formulation by branching on fractional path variables. Also, in the practical application at hand, we need to generate only very few and often disjoint paths for each commodity. From an equivalent arc-node formulation we expect to have considerably more flow variables and additional flow conservation constraints. Furthermore, the requirement of an unsplittable flow could necessitate more branching steps, possibly on several arc-flow variables for each commodity, and thereby slow down the solving process. However, whether an arc-node or a path formulation is more suitable for this problem remains to be decided by a comparative computational study.

In Section 4.7 we present heuristic methods for our path formulation that determine a suitably sparse set of paths which is sufficient for our first test instance. In Section 4.9 we focus on a heuristic path generation method that allows to solve for larger instances. But of course a rigorous comparison of some sophisticated branch-and-price algorithm that may use an arc-node formulation remains for future work. We now detail a path formulation that incorporates complex transportation tariffs.

We consider special logistics networks where the node set is composed of source nodes, sink nodes, and hub nodes. Directed arcs connect each source to every hub node, each hub to every sink node, and anti-parallel arcs connect each pair of hubs. Arcs represent transport relations where a tariff a system applies that reflects transportation cost. Each demand has to be shipped from its source to its sink node using a succession of transport relations, a path in the network.

Let $\mathcal{P}_j$ be the set of paths from a demand's source to its sink, and $\mathcal{P} = \cup_{j \in J} \mathcal{P}_j$ the union of all paths over all demands. Let $r_j^P \in \{0,1\}$ be the decision variable, which is 1 iff demand j is routed along path P. Also, denote the set of all properties (e.g. weight and volume) by

Π and let a be a vector with dimension $|\Pi|$. Then $c_e^T(a)$ is defined as the *transportation cost* for a on e, i.e., the value of an optimal solution to the Single Tariff Selection problem for edge e and property demand a.

To define this formally, let $K(e)$ be set of tariff levels (containers) available at edge e and assume that the sets $K(e)$ are disjoint for different e. In the remainder we often use $K := \cup_{e \in E} K(e)$ to shorten notation. Now, recall that each tariff level $k \in K(e)$ has a capacity $\beta_{\pi k}$ for all properties $\pi \in \Pi$ and a cost g_k. Then we define the function $c_e^T : \mathbb{R}_+^{|\Pi|} \to \mathbb{R}_+$ by

$$c_e^T(a) := \min_{k \in K(e)} (g_k \min\{m \in \mathbb{N} : m\beta_{\pi k} \geq a_\pi \ \forall \pi \in \Pi\}). \tag{4.1}$$

According to a choice of paths for each demand, let now $a(e)$ denote the resulting vector of property demands summed over all demands routed along edge e. With this notation the

deterministic problem reads as follows:

$$\text{ROUTE} \quad = \min \sum_{e\in E} c_e^T(a(e)) + \sum_{j\in J}\sum_{P\in\mathcal{P}_j} c_j^P r_j^P \tag{4.2}$$

$$\text{s.t.} \quad a(e) = \sum_{j\in J}\sum_{P\in\mathcal{P}_j : e\in P} d_j \alpha_j r_j^P \qquad \forall\, e\in E \tag{4.3}$$

$$\sum_{P\in\mathcal{P}_j} r_j^P \geq 1 \qquad \forall\, j\in J \tag{4.4}$$

$$r_j^P \in \{0,1\} \qquad \forall\, j\in J, P\in\mathcal{P} \tag{4.5}$$

In this formulation we have split the transportation costs $c_e^T(a(e))$, which can be expressed edge-wise, from path-wise costs c_j^P that comprise handling cost at intermediate hub nodes for the corresponding demand j. This distinction will become useful later.

This is a mathematical formulation for a realistic model of the deterministic, strategic planning. Note, that it is not a mixed integer linear program, as c_e^T is the solution to a subproblem. From this starting point we will derive robust and more tractable models.

4.5 Robust Model

The model presented in Section 4.4 is deterministic. It assumes fixed or *nominal* demand values $d \in \mathbb{R}_+^J$ for the original demands. In industry this data is often the average value of historical data or a forecast value. The output matrix $\mathcal{R} := (r_j^P)_{P\in\mathcal{P},\, j\in J} \in \{0,1\}^{\mathcal{P}\times J}$ of the deterministic problem specifies exactly one selected path $P \in \mathcal{P}_j$ with $r_j^P = 1$ for each demand j. We call $\mathcal{R}$ a *solution*. To such a solution we associate the collection $\{P \in \mathcal{P} \mid \exists j : r_j^P = 1\}$ of selected paths to satisfy all demands and define $E(\mathcal{R}) := \{e \in E \mid \exists\, P \in \mathcal{P}\ \exists\, j \in J : e \in P \wedge r_j^P = 1\}$ as the set of *used edges*.

We can also interpret such a solution $\mathcal{R}$ in the setting of uncertain demand values: For some realization of demand values, each occurring demand is routed along its path in $\mathcal{R}$ whereas the tariff choice on each transport relation is adjusted to the occurring demand values. Thus the cost of a solution $\mathcal{R}$ now depends on the realization of the demand values. Note, that consolidation effects and economies of scale of some solution that is optimal for fixed demand values may vanish when demand values deviate. The routing cost in a deviating scenario may be significantly higher than the computed cost. We seek robust solutions, that perform well even under their specific worst case scenario of deviating demand values. To this end we define a robust cost function that evaluates a solution by its worst case scenario cost.

We define an uncertainty set containing all scenarios of the demand values that are likely to occur in the planning horizon. We measure each solution with respect to its specific worst case scenario in the scenario set and call a solution robust optimal if it has minimum such worst case cost. This cost comes with the guarentee that it is not exceeded in any scenario of the described scenario set. Also, every other solution will in some scenario have cost higher than this cost.

4.5.1 Scenario Set for the Demand Values

We apply a model of budgeted scenario sets similar to [BS03] for the uncertain demand values. We use historical data to derive uncertainty intervals for each demand value. We assume that d_j is uncertain in the interval $[\bar{d}_j, \hat{d}_j]$, the lower bound being the nominal value. Note, that we can exclude deviations below the nominal value for a worst case approach, as transportation costs are non-decreasing in demand values and thus such deviations will not worsen the cost. If the scenarios are only restricted by these intervals, a worst case scenario for any solution is attained when all demand values are at the maximum of their interval. Obviously, protecting against such scenarios is over conservative. Thus, we restrict by $\Gamma \in \mathbb{N}$ the number of demands deviating from their nominal value in a scenario. This yields the following scenario set:

$$\{(d_j)_{j\in J} : d_j \in [\bar{d}_j, \hat{d}_j] \wedge \mid d_j \neq \bar{d}_j \mid \leq \Gamma\} \tag{4.6}$$

The parameter Γ is a user's choice of conservatism, and different values for this should be tested. Interestingly, in our case study several values for Γ gave solutions with similar cost when averaged over historical data. They however differ in the nominal and worst case costs for different parameters Γ.

It is important to stress that we restrict the absolute number of demands affected by deviations. We do not consider scenario sets where the sum of normalized deviations is bounded by a parameter. As for each edge and each demand the transportation cost is a function of the demand value with discrete jump points, these two models will in general give different worst case scenarios.

Computing a worst case scenario and corresponding worst case cost for a fixed solution is itself an optimization problem over the given scenario set, called the *Adversary Problem*.

4.5.2 The Adversary Problem

For a given solution $\mathcal{R}$ the adversary's task is to find a set of Γ-many demand values whose deviation causes the highest possible additional cost. This is itself a bi-level problem, as the cost of a deviation is the outcome of a subsequent optimization of the tariffs on every edge, i.e., the value c_e^T of the STS problem. We show in the following subsection, that the adversary problem is NP-hard. Here we formulate it as a mixed integer linear program that integrates the minimization of the cost by a subsequent choice of tariff levels into the maximization of cost by choice of deviating demand. Throughout the following sections, we set $\tilde{d}_j := \hat{d}_j - \bar{d}_j$ as the additional possible demand for j.

Modeling the Adversary Decision The adversary decisions are twofold: First, the adversary has to choose demands he wants to increase. This is done by a binary variable of type μ_j for each demand, and their sum is bounded by Γ as in the definition of the scenario set (cf. (4.6)), thus

$$\sum_{j\in J} \mu_j \;\leq\; \Gamma \tag{4.7}$$

Second, we consider an edge e that carries a flow of demands according to the current solution $\mathcal{R}$. Here for each tariff level $k \in K(e)$ the adversary must choose a property that

becomes cost driving according to cost functions c_e^T. Observe, that in the inner minimization of (4.1), we can identify one property for each tariff level k that is responsible for the multiplicity needed, while ignoring all other properties. This cost driving property is chosen by the adversary with binary variables of type $h_{k\pi}$ for each property π and each tariff level $k \in K(e)$. Inequalities (4.8) ensure at most one is selected for each tariff level. Note, that a tariff level $k \in K$ is always specific for its edge $e =: e(k)$. Thus, using k as an index implicitly also refers to an edge.

These two decisions also imply what properties of a demand are deviating on a tariff level. In our modeling we encode this with additional binary variables $h'_{kj\pi}$ deciding this property deviation. The implication is ensured with Inequalities (4.9) and (4.10) linking $h'_{kj\pi}$ to μ_j and to $h_{k\pi}$ respectively. Observe that 'binary' can be equivalently relaxed to $h'_{kj\pi} \in [0,1]$ here.

$$\sum_{\pi\in\Pi} h_{k\pi} \leq 1 \qquad \forall\, k\in K \tag{4.8}$$

$$h'_{kj\pi} \leq \mu_j \qquad \forall\, k\in K,\, j\in J,\, \pi\in\Pi \tag{4.9}$$

$$h'_{kj\pi} \leq h_{k\pi} \qquad \forall\, k\in K,\, j\in J,\, \pi\in\Pi \tag{4.10}$$

It will be become apparent later in the context of choosing the minimal tariff level, why the cost driving property $h_{k\pi}$ and the deviating property of a demand $h'_{kj\pi}$ is stored for each tariff level, independent of whether it is used or not.

The objective function The cost, that the adversary wants to maximize is, split in two parts: the involved edge cost c_e^T, and the easy path cost, i.e., handling cost. For the former we introduce variables c_e and the latter is comprised of the nominal path cost $c_j^P r_j^P$, which is a constant in the adversary problem, and deviating path cost $\tilde{c}_j^P \mu_j r_j^P$ which directly depend on the adversary decision and can be preprocessed. Recall that r_j^P encodes whether path P is chosen for demand j, and is also a constant for the adversary problem. Together we obtain two objective terms:

$$\sum_{e\in E(\mathcal{R})} c_e \;+\; \sum_{j\in J}\sum_{P\in\mathcal{P}_j} (c_j^P + \tilde{c}_j^P \mu_j) r_j^P$$

Modeling the increase in container multiplicity Since c_e^T depends on the container multiplicities necessary for each tariff level k, we now introduce several integer variables $f_{k\pi}$, one for each property π to attain the correct multiplicity in case this property is cost driving for k. Of course the adversary tries to maximize these variables, and we argue later, that for non cost driving properties $f_{k\pi}$ is forced to zero. The correct multiplicity is enforced by:

$$f_{k\pi} \leq \sum_{\substack{j:\exists P\in\mathcal{P}_j:\\ e\in P,\, r_j^P=1}} \left((\bar{d}_j \alpha_{j\pi}/\beta_{\pi k}) h_{k\pi} + (\tilde{d}_j \alpha_{j\pi}/\beta_{\pi k}) h'_{kj\pi} \right) + (1-\epsilon) h_{k\pi} \;\; \forall\, e\in E,\, k\in K(e),\, \pi\in\Pi \tag{4.11}$$

To see this, observe that demand j requires $\alpha_{j\pi}$ capacity for property π and containers of type k offer $\beta_{\pi k}$ capacity for this property. Hence, the fraction $\alpha_{j\pi}/\beta_{\pi k}$ is the fractional

multiplicity of container k required by one unit of demand j when only property π is considered. Now the sum in the right hand side of (4.11) runs over all demands that use edge $e(k)$. In the inner summands the first term adds a required fractional multiplicities for nominal demand values whereas the second term is responsible for additional multiplicities for deviating demand values, if any.

Finally the term, $(1-\epsilon)h_{k\pi}$ ensures that $f_{k\pi}$ attains the ceiling of the aforementioned sum in case $h_{k\pi}=1$. Here, ϵ can be chosen to match half the minimum occurring fractional multiplicity that a single demand requires. Also note that $h_{k\pi}=0$ with Inequalities (4.10) implies that all the terms in the former sum are zero, thus $f_{k\pi}=0$.

Modeling the minimization over the tariff levels Having established that $f_{k\pi}$ models the multiplicity of container type k when π is its cost driver, and equals zero otherwise, the minimization can be modeled in a standard way: The minimization is integrated by using the variables $f_{k\pi}$ separately for each container type k regardless of whether this type is actually used. The variable c_e shall model the new transportation cost of all demands routed along edge e. As the optimization tries to maximize this variable the Inequalities of type (4.12) push it up until the minimum cost over all tariff levels $k \in K(e)$ of edge e. Recall that g_k is the cost for one container of that tariff level.

$$c_e \;\leq\; g_k \sum_{\pi\in\Pi} f_{k\pi} \qquad\qquad \forall\, e\in E,\, k\in K(e) \tag{4.12}$$

We are now able to state the adversary problem ADV($\mathcal{R}$).

Theorem 4.1 *For a fixed solution $\mathcal{R}=(r_j^P)_{P\in\mathcal{P},\,j\in J}$ to* ROUTE *the objective value of mixed integer linear program* ADV($\mathcal{R}$) *equals the maximal cost of any scenario of the scenario set given in* (4.6).

$$\text{ADV}(\mathcal{R}) \;=\; \max \sum_{e\in E(\mathcal{R})} c_e \;+\; \sum_{j\in J}\sum_{P\in\mathcal{P}_j} (c_j^P + \tilde{c}_j^P \mu_j) r_j^P \tag{4.13}$$

$$\text{s.t.} \qquad (4.7)-(4.12)$$

$$h_{k\pi},\, \mu_j \in \{0,1\},\; h'_{kj\pi} \in [0,1], \quad c_e \;\geq\; 0,\, f_{k\pi}\in\mathbb{Z}_+ \quad \forall\, e\in E,\, k\in K,\, j\in J,\, \pi\in\Pi \tag{4.14}$$

Proof. First observe, that that for each j we can assume either $d_j=\bar{d}_j$ or $d_j=\hat{d}_j$ in a maximizing scenario. Once the adversary has invested one unit of the uncertainty budget Γ for demand j, putting it's demand value to the upper bound $\hat{d}_j$ of the interval cannot decrease transportation cost. Thus the adversary's choice of demand values is given by: $d_j=\bar{d}_j+\mu_j\tilde{d}_j$ and Inequalities (4.7) ensure $(d_j)_{j\in J}$ belongs to the set from (4.6).

For every edge e let $a(e)$ be the new vector of property demands on e resulting from deviating demand values $(d_j)_{j\in J}$. Also let π^* be the property that is cost driving for some fixed tariff level k, that is $h_{k\pi^*}=1$. In an optimum solution of ADV($\mathcal{R}$) we can then assume $h'_{kj\pi^*}=1$ iff $\mu_j=1\;\forall j$ and thus the RHS of (4.11) can be written as $a(e)_{\pi^*}/\beta_{\pi^*k}+(1-\epsilon)$. Thus the variable $f_{k\pi^*}$ attains the integer multiplicity of container type k needed to fit the old and new demand. In case, some property π is not the cost drive for k then $f_{k\pi}=0$. Also by Inequalities (4.8) at most one property per tariff level k can be cost driving. It follows,

that the RHS of (4.12) models the cost for tariff level k correctly in an optimum solution of ADV($\mathcal{R}$).

Since c_e is maximized in the objective and bounded by the minimum tariff level costs with Inequalities (4.12) it follows $c_e = c_e^T(a(e))$ and thus the value ADV($\mathcal{R}$) coincides with maximum scenario cost with respect to (4.6). □

We show next that the adversary problem is NP-hard.

4.5.3 Complexity of the Adversary Problem

For the complexity of the adversary problem we show the following theorem.

Theorem 4.2 *The adversary problem* ADV*, i.e. determining for a fixed solution* $\mathcal{R} = (r_j^P)_{P\in\mathcal{P},\, j\in J}$ *to* ROUTE *the maximal cost of any scenario in the scenario set given in* (4.6) *is* $\mathcal{NP}$*-hard.*

Proof. We show a reduction from *Set Cover*: Given a ground set Q of elements to be covered and a collection $\mathcal{S}$ of sets $S_j \subset Q$, for the decision variant of *Set Cover* we ask, whether there is a sub-collection $S_1, \cdots, S_m$, of at most m sets, such that all elements in Q are contained in at least one set. We can solve this problem by solving a suitably constructed instance of the adversary problem, in which we allow a robustness budget of $\Gamma = m$.

In this instance, we only use uncertain demand intervals of the form $[0, 1]$, which means, that the nominal cost of the solution is 0. We also set all path costs c_j^P and possible path cost increases $\tilde{c}_j^P$ to 0.

We consider a network that has only one edge but several tariff levels on this edge. All demands have their source at the start node and their sink at the end node of this edge and use it in the current solution. We introduce demands such that each one models a set in the *Set Cover* instance, so S_j can be identified with a demand $j \in J$.

We also let the set of properties Π coincide with Q, thus introducing a separate property for each ground element. The property vector of demand j is introduced as the characteristic vector of the set S_j. Finally we introduce $|Q| + 1$ different container types on our edge. The first $|Q|$ are identified with ground elements, that is, container type $k(q)$ for ground element $q \in Q$ has capacities $\beta_{k\pi} = |\mathcal{S}|$ if π is not identified with q and capacity ϵ if π belongs to ground element q. The cost factor g_k for these containers is 1. The last container with index $|Q| + 1$ has cost 2 and capacities equal to $|\mathcal{S}|$ for all properties. We claim that there is a set cover of size m iff the adversary cost is 2.

If there is a set cover of size m, then the adversary can select demands corresponding to the sets in the set cover. Thus, each property occurs in at least one selected demand. As a consequence each of the cheaper containers with index less than $|Q|+1$ requires a multiplicity f_k of $1/\epsilon$ to cover all demands resulting in higher cost than incurred by the container with index $|Q| + 1$, that can carry all demands with cost 2.

Conversely, if there is no set cover of size m, the sum of all properties of deviating demands will leave at least one property be zero, no matter what demands are selected by the adversary. Thus one of the cheaper container types can transport all deviating demands with multiplicity and cost one. □

4.5.4 Formulation of the Robust Model

By Theorem 4.2 we obtain the robust counterpart of the routing problem ROUTE by replacing the deterministic cost with the adversary cost ADV($\mathcal{R}$), that depends on the solution $\mathcal{R}$ in the multicommodity flow:

$$\text{ROBROUTE} = \min \ \text{ADV}(\mathcal{R}) \tag{4.15}$$

$$\text{s.t.} \quad \sum_{P \in \mathcal{P}_j} r_j^P \geq 1 \qquad \forall\, j \in J \tag{4.4 rev}$$

$$r_j^P \in \{0,1\} \qquad \forall\, j \in J, P \in \mathcal{P} \tag{4.16}$$

This problem still involves two levels: the routing decisions $(r_j^P)_{P\in\mathcal{P}, j\in J}$ building a current solution $\mathcal{R}$ are input for the second optimization of ADV($\mathcal{R}$), i.e., the NP-hard adversary problem. To find robust routings for instances of real-world size we adopt the following strategy: First, we linearize the tariff cost $c_e^T(a)$ in Section 4.6.1. This allows to reduce the problem to a single stage optimization (cf. Section 4.6). Second, we relax the adversary problem. Thereby, we give the adversary more power, and our solutions might be more conservative than necessary for the exact model.

Finally, once that we found a routing $\mathcal{R}$ we evaluate its robust cost with the *exact* model for the adversary problem. Though this problem is NP-hard, it can be solved in a few minutes with the mixed-integer linear program ADV($\mathcal{R}$). We report on the results of this evaluation in detail in Section 4.8.

4.6 A Solvable Model

To obtain a solvable model for finding robust routings for real-world size instances we apply two simplifications: First, we transform the edge costs c_e^T into piece-wise linear functions with a single initial jump point. Second, we give a linear relaxation of the adversary problem.

A soft argument to justify the linearization of the transportation cost to all but one jump point is provided by the uncertainty of the demand values: A routing that is good because it exploits the jump point structure, is a routing where many containers are filled exactly to capacity. Yet, as the demand may deviate such a routing is not desirable. We leave the initial jump point in the model, because this jump models the fix cost incurred by setting up a certain transport relation. We now explain these two simplification formally.

4.6.1 Simplified Tariff Cost

The goal of this subsection is to simplify the edge cost functions of type

$$c_e^T(a) = \min_{k \in K(e)} \left(g_k \min\{m \in \mathbb{N} : m\beta_{\pi k} \geq a_\pi \ \forall \pi \in \Pi\}\right)$$

while preserving the following features that are most important to keep cost low in practice:

1. The initial jump point should be kept, because it represents fixed cost for setting up a transport relation.

2. The economies of scale must still be modeled.

3. The positive effects of consolidating goods with different properties must still be modeled.

We propose a partial linearization which is natural, but has also been inspired by [Kli90] and [NP07]. Define the simplified cost $c_e^L(d) : \mathbb{R}_+^J \to \mathbb{R}_+$ now depending on a demand vector d, and not depending on a property demand vector a, as previously. The reason will become clear in the next section. First, for each tariff level k at the edge e we calculate the fractional required multiplicity $\ell_k(d)$ for this container type, maximizing over all properties in Π and summing over all demands j using the edge e:

$$\ell_k : \mathbb{R}_+^J \to \mathbb{R}_+,\ \ell_k : d \mapsto \max_{\pi\in\Pi} \sum_{j\in J(e)} \sigma_{kj\pi} d_j \tag{4.17}$$

Here, $\sigma_{kj\pi} := \alpha_{j\pi}/\beta_{\pi k}$ is the fractional multiplicity of container type k required by one unit of demand j when only property π is considered. This fractional multiplicity ℓ_k for each tariff level is used as argument for the partial linearization of c_e^L: For each tariff level at edge e we define an affine linear function linearizing the cost for using this container type:

$$c_k : \mathbb{R}_+ \to \mathbb{R}_+,\ c_k : \ell' \mapsto o_k + m_k \ell' \tag{4.18}$$

The choice of coefficients o_k and m_k are detailed in Section 4.7. Finally, we choose the container type with minimal partially linearized cost:

$$c_e^L(d) := \min_{k\in K(e)} c_k(\ell_k(d)) \tag{4.19}$$

Facility location model for tariff cost

A path solution $\mathcal{R}$ defines for each edge e a vector $d(e)$ giving the commodity flow on that edge. Note, the j-th component of $d(e)$ is defined as $d_j(e) = d_j$ if e occurs in the path P with $r_j^P = 1$ and $d_j(e) = 0$ otherwise.

Appling the partially linearized function $c_e^L(\cdot)$ to $d(e)$ we get the following mixed integer program for the simplified transportation cost on e of $\mathcal{R}$:

$$c_e^L(d(e)) = \quad \min \sum_{k\in K(e)} o_k y_k + \sum_{k\in K(e)} m_k \ell_k \tag{4.20}$$

$$\text{s.t.} \quad 1 \le \sum_{k\in K(e)} z_{kj} \qquad \forall\, j\in J \tag{4.21}$$

$$\sum_{j\in J} \sigma_{kj\pi}\, d_j(e)\, z_{kj} \le \ell_k \qquad \forall\, k\in K(e),\ \pi\in\Pi \tag{4.22}$$

$$\sum_{k\in K(e)} y_k \le 1 \tag{4.23}$$

$$y_k \ge z_{kj} \qquad \forall\, k\in K,\ j\in J \tag{4.24}$$

$$z_{kj} \in \{0,1\},\ y_k \in \{0,1\},\ \ell_k \ge 0 \qquad \forall\, k\in K,\ j\in J \tag{4.25}$$

This program is very similar to a facility location problem, when the tariff levels on edge e are interpreted as facilities and demands using e are interpreted as clients. Here binary variables z_{kj} model the assignment of commodities to tariff levels and binary variables y_k decide which tariff level $k \in K(e)$ is used to attain the minimum in Equation (4.19), or we say, which facility is opened. Then, Inequalities (4.24) ensure that this happens, as soon as some demand is assigned to it. There are two differences to the classical uncapacitated facility problem:

1. Inequality (4.23) ensures that only one tariff level, i.e., facility is chosen. However, in preliminary tests we computed $c_e^L(d(e))$ for a representative sample of edges and possible flow values $d(e)$ for test instances of the application at hand. We could observe, that removing this inequality does not change the value $c_e^L(d(e))$. In the real-world instances, the setup cost o_k of tariff levels is generally high such that there is no incentive to use more than one tariff level. Therefore we drop this equality in the remainder. Note that this does not change the practical problem significantly but it does simplify the mathematical model.

2. Instead of linear connection costs, we have costs that are linear in the maximum property usage. We model the corresponding value of the function ℓ_k from (4.17) with an identically named variable and Inequalities (4.22). Note that ℓ_k is a variable for the remainder of this chapter, while it denoted a function so far.

Despite these differences we mention the similarity explicitly, because we will use the facility location type program in the remainder, and because our model shows computationally advantageous features in the computational study that are probably an inheritance from facility location. Throughout the next sections, we also refer to tariff levels as facilities.

Combining simplified tariff cost with MCF path formulation

We now combine the simplified cost function with the path formulation of the deterministic routing problem.

We have to link the path assignment variables r_j^P to the facility assignment variables z_{kj} on each edge e. To this end we combine (4.21) and (4.3) to Inequalities (4.27), which ensures that whenever demand j is assigned to a path containing edge e, then this demand must also be assigned to some facility $k \in K(e)$ of this edge.

Constraints (4.24) now occur not only for individual sets $K(e)$ but for the set of all facilities $K := \cup_e K(e)$. Inequalities (4.22) can be uniformly formulated for all facilities $k \in K$ as Inequalities (4.28) by replacing the edge dependent demand vector $d(e)$ with the global

demand vector d and using slack variables $s_{k\pi} \geq 0$. This way we obtain:

$$\text{ROUTELIN} = \min \sum_{k\in K} o_k y_k + \sum_{k\in K} m_k \ell_k + \sum_{j\in J} \sum_{P\in\mathcal{P}_j} c_j^P r_j^P \tag{4.26}$$

$$\text{s.t.} \quad \sum_{P\in\mathcal{P}_j:\ e\in P} r_j^P \leq \sum_{k\in K(e)} z_{kj} \qquad \forall\, e\in E, j\in J \tag{4.27}$$

$$\sum_{j\in J} \sigma_{kj\pi} d_j z_{kj} + s_{k\pi} = \ell_k \qquad \forall\, k\in K, \pi\in\Pi \tag{4.28}$$

$$\sum_{P\in\mathcal{P}_j} r_j^P \geq 1 \qquad \forall\, j\in J \tag{4.4 rev}$$

$$y_k \geq z_{kj} \qquad \forall\, k\in K, j\in J \tag{4.24 rev}$$

$$r_j^P \in \{0,1\},\ z_{kj} \in \{0,1\},\ y_k \in \{0,1\},\ \ell_k \geq 0 \qquad \forall\, k\in K,\ j\in J,\ P\in\mathcal{P} \tag{4.29}$$

This is an integer linear formulation for the deterministic model with nominal demand values and partially linearized transportation cost.

4.6.2 Robust Counterpart with Relaxed Adversary

In this section we proceed towards a mixed integer linear program formulation for robust strategic planning. First, we formulate the adversary problem for the partially linearized cost of the previous section. Second, we linearize the resulting bilinear adversary problem to integrate it into ROUTELIN. The closed-form formulations can be achieved by a typical dualization technique that can be found in [BS03]. In the end, this yields a mixed integer linear program that is solvable for large instances by standard software together with a set of computational techniques, which we detail in the next section.

4.6.3 The Adversary Problem for the Simplified Tariff Cost

Different from the previous adversary problem, in this chapter we not only fix the path solution $\mathcal{R} = (r_j^P)_{P\in\mathcal{P},\, j\in J}$ but also the facility assignment matrix $Z := (z_{kj})_{k\in K,\, j\in J}$ together with all variables y_k. The first possibility for an adversary to increase cost with deviating demands d_j is to increase the maximum property usage ℓ_k in (4.28). If we again express deviating demand values with binary variables μ_j as $d_j = \bar{d}_j + \mu_j \tilde{d}_j$, we can account for this maximum property increase with a variable λ_k for each facility and obtain:

$$\lambda_k = \max_{\pi\in\Pi} \left(\sum_{j\in J} \tilde{\sigma}^*_{kj\pi} \mu_j - s_{k\pi} \right) \qquad \forall\, k\in K \tag{4.30}$$

Here we set $\tilde{\sigma}^*_{kj\pi} := \sigma_{kj\pi} \tilde{d}_j z_{kj}$ for shorter notation. Since at least one property is tight in ROUTELIN, there is at least one slack variable that is zero. So λ_k is nonnegative.

The second possibility for an adversary is to increase path cost by $\tilde{c}_j^P$ for some selected path P and a deviating demand j. Accounting for both possibilities we obtain ADVLIN$(\mathcal{R}, Z)$

which equals the maximal cost of any scenario in the scenario set given in (4.6):

$$\text{ADVLIN}(\mathcal{R}, Z) = \max \quad \sum_{k\in K} m_k\lambda_k + \sum_{j\in J}\sum_{P\in\mathcal{P}_j}(c_j^P + \tilde{c}_j^P\mu_j)r_j^P \tag{4.31}$$

$$\text{s.t.} \quad (4.30)$$

$$\sum_{j\in J}\mu_j \leq \Gamma \tag{4.34 rev}$$

$$\mu_j \in \{0,1\},\ \lambda_k \geq 0 \qquad \forall\, j\in J,\ k\in K \tag{4.32}$$

Note that as solution $(\mathcal{R}, Z)$ is given, the variables $s_{k\pi}$, z_{kj} and r_j^P are coefficients here. This formulation has the drawback, that it contains a nested maximization in (4.30) to determine λ_k. We address this with the following theorem:

Theorem 4.3 *For a fixed solution $\mathcal{R}, Z$ to* ROUTELIN*,* ADVLIN$(\mathcal{R}, Z)$ *can be obtained with a mixed integer linear program with bilinear cost function. It holds* ADVLIN$(\mathcal{R}, Z) =$

$$\max \quad \sum_{k\in K} m_k\left(\sum_{\pi\in\Pi}\sum_{j\in J}\tilde{\sigma}^*_{kj\pi}\mu_j\theta_{k\pi} - \sum_{\pi\in\Pi}s_{k\pi}\theta_{k\pi}\right) + \sum_{j\in J}\sum_{P\in\mathcal{P}_j}(c_j^P + \tilde{c}_j^P\mu_j)r_j^P \tag{4.33}$$

$$s.t.: \quad \sum_{j\in J}\mu_j \leq \Gamma \tag{4.34}$$

$$\sum_{\pi\in\Pi}\theta_{k\pi} \leq 1 \qquad \forall\, k\in K \tag{4.35}$$

$$\mu_j \in \{0,1\},\ \theta_{k\pi} \geq 0 \quad \forall\, k\in K,\ \pi\in\Pi,\ j\in J \tag{4.36}$$

Proof. We can resolve the nested maximization from (4.30) in the following way: For fixed μ_j we express λ_k in Equalities (4.30) as a solution to the following integer program:

$$\lambda_k = \max \quad \sum_{\pi\in\Pi}\theta_{k\pi}\left(\sum_{j\in J}\tilde{\sigma}^*_{kj\pi}\mu_j - s_{k\pi}\right)$$

$$\text{s.t.}: \quad \sum_{\pi\in\Pi}\theta_{k\pi} \leq 1$$

$$\theta_{k\pi} \in \{0,1\} \qquad \forall\, \pi\in\Pi$$

As the underlying matrix is totally unimodular, we can equivalently use fractional variables $\theta_{k\pi} \geq 0$. Now, because variables λ_k have non-negative cost coefficients only, we resolve the two nested linear maximization problems by substituting variables λ_k into the objective function and we obtain the desired formulation. □

The problem ADVLIN contains bilinear terms $\mu_j\theta_{k\pi}$ even though the variables associated with the input solution are considered coefficients. We next proceed to linearize this problem and integrate it into the routing.

4.6.4 Linearizing the Adversary

We now seek a linear program ADVLIN2 that is a relaxation of ADVLIN, i.e., for every solution $(\mathcal{R}, Z)$ we have ADVLIN2$(\mathcal{R}, Z) \geq$ ADVLIN$(\mathcal{R}, Z)$. Thus, using ADVLIN2 for the final

formulation of a robust routing problem, we use an adversary which is not less powerful than originally envisioned. Consider the following linear program:

$$\text{ADVLIN2}(\mathcal{R}, Z) \;=\; \max \sum_{k\in K} m_k \left(\sum_{\pi\in\Pi} \sum_{j\in J} \tilde{\sigma}^*_{kj\pi} \chi_{kj\pi} - \sum_{\pi\in\Pi} s_{k\pi} \theta_{k\pi} \right) + \sum_{j\in J} \sum_{P\in\mathcal{P}_j} (c^P_j + \tilde{c}^P_j \mu_j) r^P_j \tag{4.37}$$

$$\begin{aligned}
\text{s.t. :} \qquad \sum_{j\in J} \mu_j &\;\le\; \Gamma && && \text{(4.34 rev)}\\
\mu_j &\;\le\; 1 && \forall\, j\in J && (4.38)\\
\sum_{\pi\in\Pi} \theta_{k\pi} &\;\le\; 1 && \forall\, k\in K && \text{(4.35 rev)}\\
\sum_{\pi\in\Pi} \chi_{kj\pi} &\;\le\; \mu_j && \forall\, k\in K,\, j\in J && (4.39)\\
\chi_{kj\pi} &\;\le\; \theta_{k\pi} && \forall\, k\in K,\, j\in J,\, \pi\in\Pi && (4.40)\\
\mu_j,\, \theta_{k\pi},\, \chi_{kj\pi} &\;\ge\; 0 && \forall\, k\in K,\, \pi\in\Pi,\, j\in J && (4.41)
\end{aligned}$$

Theorem 4.4 *For every solution* $(\mathcal{R}, Z)$ *to* ROUTELIN *we have*

$$\text{ADVLIN2}(\mathcal{R}, Z) \geq \text{ADVLIN}(\mathcal{R}, Z).$$

Proof. First, we free the integer variable μ_j to be fractional, i.e., take values in $[0, 1]$. This clearly is a relaxation. To eliminate the bilinear terms in the objective we replace them by variables $\chi_{kj\pi} := \mu_j \theta_{k\pi}$ and in a first step add $\chi_{kj\pi} = \mu_j \theta_{k\pi}$ to the formulation. So far this replacement leaves the problem equivalent. Observe that with μ_j and $\theta_{k\pi}$ in $[0, 1]$ we have:

$$\chi_{kj\pi} = \mu_j \theta_{k\pi} \leq \min\{\mu_j,\, \theta_{k\pi}\}.$$

Thus instead of forcing equality, the bounding of $\chi_{kj\pi}$ from above both by μ_j and $\theta_{k\pi}$ is a relaxation. As Inequalities (4.35) imply

$$\sum_{\pi\in\Pi} \chi_{kj\pi} = \sum_{\pi\in\Pi} \mu_j \theta_{k\pi} \leq \mu_j$$

we can replace the bounding by μ_j by the stronger bound over the sum as in Inequalities (4.39) and still have a relaxation. □

Example 4.1 *The relaxation* ADVLIN2 *may introduce a relative error of at least 5/4. We construct an instance and two solutions* $(\mathcal{R}_1, Z_1), (\mathcal{R}_2, Z_2)$ *with* $\text{ADVLIN}(\mathcal{R}_1, Z_1) = \text{ADVLIN}(\mathcal{R}_2, Z_2)$ *but* $\text{ADVLIN2}(\mathcal{R}_1, Z_1) = (5/4)\,\text{ADVLIN2}(\mathcal{R}_2, Z_2)$*. Consider a network with exactly one common source, one common sink, one edge from source to sink, two properties, two containers, and three demands.*

We specify the properties for demands $\alpha_{j\pi}$ *and container capacities* $\beta_{\pi k}$ *in vector notation, such that:*

$$\alpha_{1\cdot} = \begin{pmatrix} 2 \\ 2 \end{pmatrix},\; \alpha_{2\cdot} = \alpha_{3\cdot} = \begin{pmatrix} 2 \\ 0 \end{pmatrix},\; \beta_{\cdot 1} = \begin{pmatrix} 2 \\ 1 \end{pmatrix},\; \beta_{\cdot 2} = \begin{pmatrix} 2 \\ 2 \end{pmatrix},\; m_1 = m_2 = 1,$$

Nominal demand values $\bar{d}_j$ are zero and deviating values $\tilde{d}_j$ are one for all demands and we impose a robustness budget $\Gamma = 2$. We also assume all other parameters are set to zero. Now in $(\mathcal{R}_1, Z_1)$ all demands are assigned to the first tariff level, whereas $(\mathcal{R}_2, Z_2)$ only uses the second tariff level.

One can check all integral choices of μ and observe $\text{ADVLIN}(\mathcal{R}_1, Z_1) = \text{ADVLIN}(\mathcal{R}_2, Z_2) = 2$. *Now, among optimal solutions for* $\text{ADVLIN2}(\mathcal{R}_1, Z_1)$ *and* $\text{ADVLIN2}(\mathcal{R}_2, Z_2)$ *let us consider those, where $\mu_1 = 1$, $\mu_2 = \mu_3 = 0.5$ and $\theta_{11} = \theta_{12} = \theta_{21} = \theta_{22} = 0.5$.*

The parameters $\tilde{\sigma}^*_{kj\pi}$ according to previous definitions $\tilde{\sigma}^*_{kj\pi} = \sigma_{kj\pi}\tilde{d}_j z_{kj} = (\alpha_{j\pi}/\beta_{\pi k})\tilde{d}_j z_{kj}$ are given in Table 4.1 together with resulting optimal choices for χ. Summing up the corresponding contribution to the objective function (fourth and 7th column), we obtain $\text{ADVLIN2}(\mathcal{R}_1, Z_1) = 2.5$ *whereas* $\text{ADVLIN2}(\mathcal{R}_2, Z_2) = 2$.

Table 4.1: Example with $\text{ADVLIN}(\mathcal{R}_1, Z_1) = \text{ADVLIN}(\mathcal{R}_2, Z_2)$ but $\text{ADVLIN2}(\mathcal{R}_1, Z_1) = (5/4)\,\text{ADVLIN2}(\mathcal{R}_2, Z_2)$

	$(\mathcal{R}_1, Z_1)$ using container $k = 1$			$(\mathcal{R}_2, Z_2)$ using container $k = 2$		
J	$\tilde{\sigma}^*$	χ in ADVLIN2	obj.	$\tilde{\sigma}^*$	χ in ADVLIN2	obj.
$j = 1$	$\tilde{\sigma}^*_{11\cdot} = \begin{pmatrix}1\\2\end{pmatrix}$	$\chi_{11\cdot} = \begin{pmatrix}0.5\\0.5\end{pmatrix}$	1.5	$\tilde{\sigma}^*_{21\cdot} = \begin{pmatrix}1\\1\end{pmatrix}$	$\chi_{11\cdot} = \begin{pmatrix}0.5\\0.5\end{pmatrix}$	1
$j = 2$	$\tilde{\sigma}^*_{12\cdot} = \begin{pmatrix}1\\0\end{pmatrix}$	$\chi_{22\cdot} = \begin{pmatrix}0.5\\0\end{pmatrix}$	0.5	$\tilde{\sigma}^*_{22\cdot} = \begin{pmatrix}1\\0\end{pmatrix}$	$\chi_{22\cdot} = \begin{pmatrix}0.5\\0\end{pmatrix}$	0.5
$j = 3$	$\tilde{\sigma}^*_{13\cdot} = \begin{pmatrix}1\\0\end{pmatrix}$	$\chi_{23\cdot} = \begin{pmatrix}0.5\\0\end{pmatrix}$	0.5	$\tilde{\sigma}^*_{23\cdot} = \begin{pmatrix}1\\0\end{pmatrix}$	$\chi_{23\cdot} = \begin{pmatrix}0.5\\0\end{pmatrix}$	0.5

4.6.5 A Robust, Mixed Integer Linear Formulation for the Strategic Planning Problem

As we obtained a linear program for the relaxed adversary, we can integrate this in a standard way into the logistic routing problem by dualization. The dual of ADVLIN2 reads as follows:

$$\text{ADVDUAL}(\mathcal{R}, Z) = \min \quad \Gamma\omega + \sum_{j\in J}\delta_j + \sum_{k\in K}\phi_k + \sum_{j\in J}\sum_{P\in\mathcal{P}_j} c_j^P r_j^P \tag{4.42}$$

$$\omega + \delta_j - \sum_{k\in K}\gamma_{kj} \geq \sum_{P\in\mathcal{P}_j}\tilde{c}_j^P r_j^P \qquad \forall\, j\in J \tag{4.43}$$

$$\phi_k - \sum_{j\in J}\xi_{kj\pi} \geq -m_k s_{k\pi} \qquad \forall\, k\in K,\ \pi\in\Pi \tag{4.44}$$

$$\gamma_{kj} + \xi_{kj\pi} \geq m_k\tilde{\sigma}^*_{kj\pi} \qquad \forall\, k\in K,\ j\in J,\ \pi\in\Pi \tag{4.45}$$

$$\omega,\ \delta_j,\ \phi_k,\ \gamma_{kj},\ \xi_{kj\pi} \geq 0 \qquad \forall\, k\in K,\ \pi\in\Pi,\ j\in J \tag{4.46}$$

This dual formulation is again a minimization problem such that we are able to integrate it into ROUTELIN to give a closed-form mixed integer programming formulation for the robust counterpart:

ROB-LIN =

$$\min \quad \sum_{k\in K} o_k y_k + \sum_{k\in K} m_k \ell_k + \Gamma\omega + \sum_{j\in J} \delta_j + \sum_{k\in K} \phi_k + \sum_{j\in J} \sum_{P\in\mathcal{P}_j} c_j^P r_j^P \tag{4.47}$$

s.t.

$$\sum_{P\in\mathcal{P}_j:\ e\in P} r_j^P \ \le \ \sum_{k\in K(e)} z_{kj} \qquad \forall\ e\in E, j\in J \tag{4.27 rev}$$

$$\sum_{j\in J} \sigma_{kj\pi} d_j z_{kj} + s_{k\pi} = \ell_k \qquad \forall\ k\in K, \pi\in\Pi \tag{4.28 rev}$$

$$\sum_{P\in\mathcal{P}_j} r_j^P \ \ge \ 1 \qquad \forall\ j\in J \tag{4.4 rev}$$

$$y_k \ \ge \ z_{kj} \qquad \forall\ k\in K, j\in J \tag{4.24 rev}$$

$$\omega + \delta_j - \sum_{k\in K} \gamma_{kj} \ \ge \ \sum_{P\in\mathcal{P}_j} \tilde{c}_j^P r_j^P \qquad \forall\, j\in J \tag{4.43 rev}$$

$$\phi_k - \sum_{j\in J} \xi_{kj\pi} \ \ge \ -m_k s_{k\pi} \qquad \forall\, k\in K,\ \pi\in\Pi \tag{4.44 rev}$$

$$\gamma_{kj} + \xi_{kj\pi} \ \ge \ m_k \sigma_{kj\pi} \tilde{d}_j z_{kj} \qquad \forall\, k\in K,\ j\in J,\ \pi\in\Pi \tag{4.45 rev}$$

$$\omega,\ \delta_j,\ \phi_k,\ \gamma_{kj},\ \xi_{kj\pi} \ \ge \ 0 \qquad \forall\, k\in K,\ \pi\in\Pi,\ j\in J \tag{4.45 rev}$$

$$r_j^P \in \{0,1\},\ \ z_{kj}\in\{0,1\}, y_k\in\{0,1\},\ \ \ell_k, s_{k\pi}\ge 0 \quad \forall\, k\in K,\ \pi\in\Pi,\ j\in J,\ P\in\mathcal{P} \tag{4.48}$$

This model has exponentially many path variables. We show in the next section among other tricks how this can be overcome for realistic instances.

4.7 Computational Techniques

In the preceding section we obtained a model that is solvable with mixed integer programming. Note that this requires the development of a branch-and-price algorithm to deal with the paths set of exponential size. As real-world instances are large especially in terms of demands they require additional sparsity of the variable sets which prohibits a straightforward application. The main concern of this chapter is to present methods to decide before the solving process, which of the variables are less relevant and may be omitted in the formulation. The techniques presented here are heuristic, that is, we may exclude variables necessary to obtain optimum solutions. They however allow to solve problems of moderate size and a restricted set of paths, even without applying a more involved branch-and-price algorithm.

At first, we propose a heuristic estimate for transportation cost caused by single demands on each edge, that is based on upper bounds of possible commodity flow on this edge. With this estimate we can decide which edges are unlikely to occur in a demand's paths in optimum solutions and we can omit corresponding path variables.

4.7.1 Heuristic Cost Estimates

Consider a fixed demand vector $d = d(e) \in \mathbb{R}_+^J$ that represents some flow on edge e. We want to define a mechanism for sharing the cost $c_e^T(a(e))$, that is incurred at edge e for the aggregated properties $a(e) \in \mathbb{R}_+^\Pi$ with $a(e) = a(d(e)) = \sum_{j \in J} \alpha_j d_j$ (see also Section 4.4). Each demand j, with $d_j > 0$ has to pay a certain part. We propose a cost share that depends on the ratios arising for each property π from individual property usage of a single demand $d_j \alpha_{j\pi}$ as well as on the aggregated property usage of all demands in $a_\pi(d)$.

Furthermore, we want to weight each property ratio with a parameter t_π expressing how tight the aggregated property usage is with respect to a certain container type. In the following, we fix an optimum container type k^* together with multiplicity f_{k^*} that is obtained when optimizing $c_e^T(a(e))$, and we express the cost share only with respect to this container type. Let the *property tightness* be defined as $t_\pi = a_\pi(d)/f_{k^*}\beta_{\pi k^*}$ and the *weighted volume cost share* $c_e^V(j,d)$ of demand j with respect to demand values d on edge e as:

$$c_e^V(j,d) := g_{k^*} f_{k^*} \Big(\sum_{\pi \in \Pi} \alpha_{j\pi} d_j / f_{k^*} \beta_{\pi k^*} \Big) \Big/ \Big(\sum_{\pi \in \Pi} t_\pi \Big) \tag{4.49}$$

To see that this is in fact a cost share, we observe:

$$\sum_{j \in J} c_e^V(j,d) = g_{k^*} f_{k^*} \Big(\sum_{\pi \in \Pi} \sum_{j \in J} \alpha_{j\pi} d_j / f_{k^*} \beta_{\pi k^*} \Big) \Big/ \Big(\sum_{\pi \in \Pi} t_\pi \Big) = g_{k^*} f_{k^*} \sum_{\pi \in \Pi} t_\pi / \sum_{\pi \in \Pi} t_\pi = c_e^T(d)$$

Writing the inner summands of the sum in the numerator in Expression (4.49) equivalently as $t_\pi(\alpha_j d_j / a_\pi(d))$ allows the interpretation of $c_e^V(j,d)$ as a share based on the sum of volume shares $\alpha_j d_j / a_\pi(d)$ weighted with the property tightness t_π.

4.7.2 Refining Flow Bounds on Hub-incident Edges

This cost sharing mechanism is now used to exclude some of the candidate paths for a demand. By considering the remaining possible paths we can flag each arc with the demands that are allowed to be routed over this edge, or slightly more general, we maintain for each edge a demand vector $D(e) \in \mathbb{R}_+^J$ of upper bounds of possible flow on edge e. We start with the trivial bounds that only restrict flow on source incident edges. For some source were a set of demands $J' \subset J$ originates and an outgoing edge e, we set $D_j(e) = d_j$ if $j \in J'$ and $D_j(e) = 0$ otherwise. An analogue restriction is made for sinks. For all other edges $D_j(e)$ is set to the maximum possible value. These initial flow bounds can then be further refined.

The main idea for a refinement is to forbid the routing of demands along hubs that are "too far away" to be included in a cost efficient route. However, distances are neither input of the problem nor the right measure for this task, since even long detours can be beneficial for small values of demand. We propose to consider a tradeoff between the heuristic cost share from above and the direct shipping cost.

For demand j we estimate the cost $\mathrm{Dist}(H)$ to hub H from source and sink by computing cheapest paths '$\mathrm{source}_j \rightsquigarrow H$' and '$H \rightsquigarrow \mathrm{sink}_j$' with edge cost shares $c_e^V(j, D(e))$. We also compute direct shipping costs $c^{\mathrm{DIR}}(j)$ as a cheapest path '$\mathrm{source}_j \rightsquigarrow \mathrm{sink}_j$' with cost function $c_e^T(d_j \alpha_j)$. If $\mathrm{Dist}(H)$ exceeds $c^{\mathrm{DIR}}(j)$, we exclude hub H from possible routings by setting $D_j(e)$ to zero for all edges e that are incident to H.

4.7.3 Restricting Paths and Facilities

After refining flow bounds, the set of possible source-sink paths $\mathcal{P}_j$ may still have size exponential in the input of the problem instance. In practice however, additional costs and delays for handling commodities at hubs impose a small bound on the number of intermediate hub nodes in relevant routes. Clearly, for a small, fixed constant M we can generate all paths with at most M edges that respect the known flow bounds $D(e)$. But this still involves a large number of variables z_{kj} for the assignment of demand paths to the facilities of an edge. In the following we present a method to also reduce the number of these variables.

For a clearer presentation we introduce the notion of a *facility path* for a succession of facilities $k \in K$ such that the underlying edges build a source-sink path for demand j. Note that such paths are *not* explicitly generated for the MILP formulations. For some edge path $P \in \mathcal{P}_j$, $P = e_1, e_2, \ldots, e_n$ let $\mathcal{U}(P) = K(e_1) \times K(e_2) \times \ldots \times K(e_n)$ be the set of all possible facility paths induced by P. Clearly not all such facility paths contribute a cost efficient routing alternative for demand j.

We extend the idea of a *weighted volume cost share* from Subsection 4.7.1 to estimate the cost of a facility path. As a result we can exclude many such paths and also drop corresponding variables z_{jk}, related constraints and dual variables.

Recall from Definition 4.18 the function c_k expressing the facility costs and from Definition 4.17 the function ℓ_k for the maximum property usage of facility k. The previous notion for property tightness can be adapted for the usage of facility k as

$$t_\pi := \sum_j \sigma_{kj\pi} d_j / \max_{\pi'} \sum_j \sigma_{kj\pi'} d_j$$

and the corresponding weighted volume cost share is then:

$$c_k^V(j,d) := c_k(\ell_k(d)) \Big(\sum_{\pi \in \Pi} d_j \sigma_{kj\pi} / \ell_k(d) \Big) \Big/ \Big(\sum_{\pi \in \Pi} t_\pi \Big)$$

Thus, we can iterate over all facility paths $F \in \mathcal{U}(P)$ and check if the cost estimate $\sum_{k \in F} c_k^V(j,d)$ for demand j exceeds a certain cost bound, e.g. the direct shipping cost $c^{\mathrm{DIR}}(j)$ and discard unsuitable paths. Finally we only keep assignment variables of demand j to facility k if needed by at least one path.

4.7.4 From Containers to Facilities

In the Subsection 4.6.1 we introduced simplified cost functions $c_e^L(d)$, that we use to approximate exact tariff costs. We now detail how the coefficients can be obtained from the original container types. We recall the previous definitions of the cost model:

$$\ell_k : \mathbb{R}_+^J \to \mathbb{R}_+ : d \to \max_{\pi \in \Pi} \sum_{j \in J} \sigma_{kj\pi} d_j \tag{4.17 rev}$$

$$c_k : \mathbb{R}_+ \to \mathbb{R}_+ : \ell \to o_k + m_k \ell \tag{4.18 rev}$$

$$c_e^L(d) := \min_{k \in K(e)} c_k(\ell_k(d)) \tag{4.19 rev}$$

We need to choose coefficients o_k and m_k such that $c_e^L(d)$ is a good approximation of $c_e^T(a(d))$. We observe that container cost for a fixed type k can be expressed by a step cost function: $d \mapsto g_k f_k(d)$ where $f_k(d) \in \mathbb{Z}_+$ is the needed multiplicity: $f_k(d) := \max_{\pi \in \Pi} \lceil a_\pi(d)/\beta_{\pi k} \rceil$.

For each Container k we keep the initial step by setting $o_k = g_k$, while all other steps are linearized. We compute the multiplicity $f_k^{\max} = f_k(D(e))$, that is needed to cover the upper bounds of possible commodity flow $D(e)$ on this edge and set the linear cost factor m_k such that $c_k(f_k^{\max})$ equals the maximum step cost $g_k f_k^{\max}$, i.e. $m_k = g_k - g_k/f_k^{\max}$.

4.7.5 Aggregating Demands

The large number of demands—easily reaching 10,000 in some of the case studies considered—may lead to computationally intractable instances. However, many source-sink pairs have many demands in common, and economies of scale are a natural incentive to route these demands along a common path.

We want to reduce the number of demands by aggregating all demands sharing the same source and sink into a new demand called *super demand*. The non-aggregated demands are then referred to as *original* demands. Note, that some aggregated demand may in fact be a copy of some original Demand if either its source or sink happens to be exclusive to him. Aggregating demands is a restriction of the problem since splitting such a bundle into different paths could allow for beneficial consolidation on different edges.

More formally, let $\bar{J}$ be the set of all newly introduced super demands and for one such super demand $j \in \bar{J}$ let $J(j)$ be the set of all original demands that are aggregated into super demand j. The *aggregated properties* of super demand j with respect to nominal demand values $\bar{d}$ are then defined as

$$\bar{\varrho}_{j\pi} := \sum_{j' \in J(j)} \bar{d}_{j'} \alpha_{j'\pi} \, .$$

We can now optimize in the deterministic setting with super demands simply by requiring to ship one unit of each super demand. Since we aggregated with respect to nominal demand values, shipping aggregated property vectors $\bar{\varrho}_j$ produces exactly the same costs as optimizing for the original demands for nominal demand values (subject to restriction of shipping them jointly). However, this becomes more complicated in the robust setting: A straightforward idea is to aggregate the decisions about which of the original demands deviate, into one decision for each super demand. The decision, whether super demand j deviates, is then weighted with $|J(j)|$ for the robustness budget. In this setting the uncertainty set (4.6) for demand values from Section 4.5 translates into an uncertainty set for property vectors as follows:

$$\{(\varrho_j)_{j \in \bar{J}} : \varrho_j = \bar{\varrho}_j + \mu_j \sum_{j' \in J(j)} \tilde{d}_{j'} \alpha_{j'}, \, \mu_j \in \{0,1\} \wedge \sum_{j \in \bar{J} : \varrho_j \neq \bar{\varrho}_j} |J(j)| \leq \Gamma\}. \tag{4.50}$$

Here the decision variable μ_j decides whether the super demand j deviates, and in that case, all additional demand values $\tilde{d}_{j'}$ of original demands $j' \in J(j)$ contribute to the deviating properties. That means, that if the adversary selects j, then all of the original demands in $J(j)$ are selected.

This is a restriction of the adversary because selecting only some of the original demands leaves more of the robustness budget for other super demands and can lead to worse costs.

In the following we refine this idea by introducing a parameter G_j that fixes the number of possibly deviating original demands of some super demand, e.g. in our computational studies we set $G_j = \max\{1, \lceil (2/3)|J(j)| \rceil\}$. Further we count only G_j in the budget, i.e. require $\sum_{j:\varrho_j \neq \bar{\varrho}_j} G_j \leq \Gamma$. On the one hand this is a further restriction, because the adversary can no longer select all original demands of a super demand. On the other hand it gives him the power, to select different original demands to generate different property vectors of some super demand and include more super demands in a worst case scenario.

4.7.6 Uncertain Properties for Aggregated Demands

Formally, for a fixed parameter G_j the uncertain property vector ϱ_j of super demand j may realize in the set $A_j(G_j)$ defined by:

$$A_j(G_j) := \{\bar{\varrho}_j + \sum_{j' \in J(j)} \mu_{j'} \tilde{d}_{j'} \alpha_{j'} \;\Big|\; \sum_{j' \in J(j)} \mu_{j'} \leq G_j, \mu_{j'} \in \{0,1\}\}. \tag{4.51}$$

This yields the uncertainty set for uncertain property vectors of all super demands:

$$\{(\varrho_j)_{j \in \bar{J}} \;\Big|\; \varrho_j \in A_j(G_j), \sum_{j \in \bar{J}, \varrho_j \neq \bar{\varrho}_j} G_j \leq \Gamma\} \tag{4.52}$$

By this definition, whenever some super demand j deviates, G_j of the robustness budget Γ is consumed and G_j many of j's original demands can be selected to deviate. In a worst case scenario the adversary will use this degree of freedom to increase the entries in the property vector ϱ_j to maximize tariff costs over all edges on the super demand's path.

This refined uncertainty set cannot easily be integrated into the robust model from Section 4.6. We thus propose a further relaxation: Let $\tilde{\varrho}_j$ be the vector of additional worst case property values of vectors from $A_j(G_j)$, that is $\tilde{\varrho}_{j\pi} := \max\{v_\pi - \bar{\varrho}_{j\pi} \;\big|\; v \in A_j(G_j)\}$. We replace $A_j(G_j)$ with the set

$$B_j(G_j) := \{\bar{\varrho}_j + \delta \tilde{\varrho}_j \;\big|\; \delta \in \{0,1\}\}. \tag{4.53}$$

To see that this is a relaxation in our model, observe that for a fixed set J^{ADV} of adversarial demands it holds

$$\max_{\substack{a_j \in A_j(G_j) \\ \forall\, j \in J^{\mathrm{ADV}}}} \max_{\pi \in \Pi} \sum_{j \in J^{\mathrm{ADV}}} a_{j\pi} \leq \max_{\substack{b_j \in B_j(G_j) \\ \forall\, j \in J^{\mathrm{ADV}}}} \max_{\pi \in \Pi} \sum_{j \in J^{\mathrm{ADV}}} b_{j\pi}. \tag{4.54}$$

So using the uncertainty set

$$\{(\varrho_j)_{j \in \bar{J}} \;\Big|\; \varrho_j \in B_j(G_j), \sum_{j \in \bar{J}, \varrho_j \neq \bar{\varrho}_j} G_j \leq \Gamma\} \tag{4.55}$$

is indeed a relaxation in our model, because only maxima of properties such as the inner maxima in (4.54) occur when calculating worst case cost.

This relaxation lets us easily integrate interval uncertainty for original demands into the model from Section 4.6. By preprocessing the entries in $\tilde{\varrho}_j$, for example in Inequalities (4.11), we can simply replace the coefficients $\tilde{d}_j \alpha_{j\pi}$ by coefficients $\tilde{\varrho}_{j\pi}$.

4.7.7 Preprocessing Direct Connections

After aggregating demands by common source-sink pairs the transportation cost on direct connections depends on binary decisions only: First, whether the only possible super demand is routed directly or not and second, if so, whether it is included in a worst case scenario or not.

With a preprocessing of the exact cost for all these cases, we can model these decisions with a separate path $P \in \mathcal{P}j$ for each direct connections that has modified nominal and deviating cost coefficients: $\bar{c}_{Pj}$ accounts for the direct connection case with nominal demand values and $\tilde{c}_{Pj}$ for the additional cost when this super demand is selected for by the adversary. Thus we can drop all variables necessary to model tariff cost on direct connection edges and also drop corresponding inequalities.

4.8 Computational Study

In this section we evaluate the model and the solution techniques presented in the preceding sections on our running example, the inbound logistics network of an automobile manufacturer. The data was provided by our project partner 4flow AG [4fl17], a logistics consultancy company serving small, medium-sized and global customers from a broad spectrum of industries. The first test instances, named H_Auto_1 has 23 sinks, 400 sources, 5 possible hubs, and more than 10^4 demands, that could be aggregated to 754 super demands. Also three properties were relevant, mass, volume and loading meters.

4.8.1 Method of Evaluation

We solve the aggregated model and choose the local uncertainty budget parameters G_j as 2/3 of $|J(j)|$, the number of demands aggregated into a super demand j. We generate all paths variables for paths of length 2 and apply all computational techniques mentioned in Section 4.7.

We use historical data on observed weekly demand values to derive uncertainty values by taking the average demand value as the nominal lower bound and the maximum observed value as an upper bound. Note that in our case studies, a weekly delivery is required to fulfill service level requirements.

We iterate different robustness budgets Γ in $\{0, 2, 4, 8, 16, 32\}$ % of the original demands and evaluate each obtained solution by its specific worst case cost, again obtained for all different values of Γ. But this time, we use the exact adversary problem ADV for aggregated demands from Section 4.5. We report the cost for nominal demand values as well as the exact, averaged cost obtained for each of the 28 observed weeks.

All algorithms have been implemented in C++ and compiled with gcc 4.8.1 on openSUSE 13.1 Linux with kernel 3.11.10-17. Computations have been performed on cluster nodes with two DualCore-Opteron 2218 processors (2.6 GHz, 64 bit) and 16 GB of memory using CPLEX 12.5 for mixed integer programs. A time limit of 2h was sufficient for the instance in this section. For all MIP computations we impose an relative optimality gap of 0.2% and all runs terminated within this threshold unless reported otherwise.

4.8.2 Price of Robustness

We report the results of our computations in Table 4.2. Each row refers to a single solution and contains cost values for different evaluation criteria. For each evaluation criterion the best solution is emphasized.

	evaluation criteria						
Solution	nominal cost	historical average cost	worst-case-cost for Γ =2%	worst-case-cost for Γ =4%	worst-case-cost for Γ =8%	worst-case-cost for Γ =16%	worst-case-cost for Γ =32%
deterministic solution ($\Gamma = 0$)	**331,394**	**355,293**	557,679	632,469	717,239	782,579	850,958
robust solution for Γ =2%	343,877	356,354	552,541	619,382	700,355	764,527	834,572
robust solution for Γ =4%	344,528	356,437	**551,919**	**617,924**	698,447	762,507	831,144
robust solution for Γ =8%	350,502	356,295	556,014	622,121	691,155	752,916	823,366
robust solution for Γ =16%	351,939	357,297	557,421	621,797	**691,022**	**752,773**	**820,117**
robust solution for Γ =32%	364,425	361,459	569,162	633,415	699,857	762,766	824,325

Table 4.2: Robust and deterministic solutions for H_Auto_1

The solution with $\Gamma = 0$, in the following phrased *deterministic solution* achieves the best value for nominal costs. The other solutions achieve worse nominal costs: the higher the parameter Γ is chosen for optimizing a solution, the higher is its nominal cost. The most conservative solution ($\Gamma = 32\%$) is up to 10% more expensive than the deterministic one for nominal demand. When looking at the historical average however this difference shrinks to 2% because the average cost for the deterministic solution is much higher than its nominal cost.

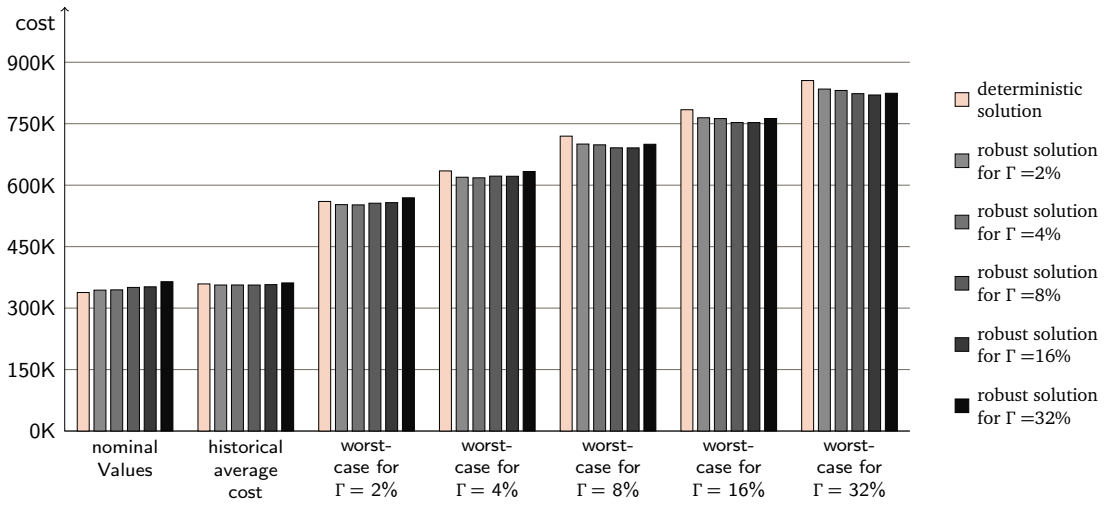


Figure 4.1: Robust and deterministic solutions for H_Auto_1

For the robust solution, one would expect to find the best marked entry on the diagonal of

Table 4.2 because here the robustness budget used during optimization and for the evaluation criterion coincide. The reasons that this is not the case can be found in the linearization and relaxation steps necessary to solve these models. However, when looking at Figure 4.1, that visualizes the results from Table 4.2, we can observe that optimizing for a certain budget Γ, also pays off for the Γ-worst case cost.

A remarkable observation is, that already a small percentage of 8 % of deviating demands can lead to an enormous cost increase of more than 100 % over the historical average. Immunizing against such deviations can save up to 3.5% of the worst case cost. At the same time, all solutions computed, except the one optimized for 32% perform roughly equally on the historical average.

4.9 Larger Instances

The instance considered in our case study, H_Auto_1, is a real instance of an automotive supplier and quite large. In this section we investigate ways to solve even larger instances. An example for a very large instance is provided by H_Auto_4 with up to 1800 aggregated demands, 12 possible hub nodes, 1,412 sources, and 14 sinks. For such instances the model formulations developed in this chapter exhibit a huge number of variables—roughly half a million for H_Auto_4, even when using the techniques from Section 4.7. This prevents state of the art MIP software from solving even the linear relaxation in acceptable running time. In this section we present two techniques that allow to solve even such very large scale instances. Note, that the ideas in this section serve as an outlook and have proof of concept status.

4.9.1 Clustering the set of commodities

The first technique is a clustering of the demand set J into clusters of demands $J_1, \ldots, J_b$. We solve the routing instances for these clusters as individual sub instances independently. A final solution is obtained by adding the flow patterns of the sub-instances and re-optimizing the tariff selection for the resulting flow.

When solving one of the sub-instances, we loose the optimization potential of intentionally consolidating flow with demands that belong to different clusters. In the robust setting, we also subdivide the uncertainty budget $\Gamma =: \Gamma_1 + \ldots + \Gamma_b$ into uncertainty budgets for each partition according to the cardinality of the partition. This is clearly a restriction of the adversary problem of the unpartitioned instance, since the set of Γ worst demands could fall into the same cluster, but the uncertainty budget for that cluster only allows to select a subset of these of size J_j. To keep the effects of these errors low in practice, we choose the partitioning of the demands carefully. We now briefly describe the method used for this partitioning.

We define a graph $\mathcal{K}$ whose node set is the set of demands J and an edge between two demands expresses the likeliness of these two demands being routed together in near optimal solutions. We seek a partition of this graph into b clusters of roughly equal size, such that the weight of the total cut, that is the sum of all edges with end nodes in different clusters, is minimized. This minimization should reduce the loss in optimization potential and also in the power of the adversary.

We propose the following measure: let $\overrightarrow{w}(j,u)$ be the minimum cost for transporting demand j from its source to node u using the exact tariff cost function c^T from Section 4.4 for

nominal demand values and ignoring any possibilities of consolidation with other demands. Conversely $\overleftarrow{w}(j,u)$ is the corresponding cost for transportation from u to its sink.

Now for two demands $i, j \in J$ we set the edge weight $\omega(i,j)$

$$\omega(i,j) := 1/\left(\min_u \overrightarrow{w}(i,u) + \overrightarrow{w}(j,u) + \min_v \overleftarrow{w}(i,v) + \overleftarrow{w}(j,v)\right).$$

The denominator accounts for the cost for transporting both demands from their sources to a common node u and from another common node v to their sinks. Taking the reciprocal value here ensures that demands with close common nodes u and v have a high edge weight. Finally, we use the algorithmic toolkit from [SS13] to compute a partition minimizing ω over the total cut induced by the clusters.

4.9.2 Generating paths and tariff levels

In the computational study we observe that the linear programming (LP) relaxation of formulation ROB-LIN is strong. Solving this relaxation is however difficult. Most of the variables are needed to model the assignment of demands to paths in the network and for modeling the complex cost structure on the edges. In this section we propose a method that leaves out most of these variables and generates them iteratively.

We do not pursue an exact branch-and-price approach but confine ourselves to a simpler non-exact method which satisfies the requirements of our industrial partner. Though it looks similar to a standard branch-and-price algorithm (for an overview see [DDS05]), it is only heuristic for two reasons: First we do not solve the pricing problem for the underlying LP relaxation exactly. Since in our setting, two type of variables have to be generated in one pricing step, an exact solution method to the pricing problem is non-trivial. Second, we stop the generating approach after a small number of rounds. Finally, we solve the formulation with all variables generated so far as a mixed integer program by a standard solver. In contrast to branch-and-price we do not apply column generation at each node in the branch and bound tree.

As mentioned above, the study here does not aim at a competitive implementation for these instances but is only a proof of concept that shows that the model introduced in this chapter is applicable to even larger instances when combined with techniques like column generation. We nevertheless briefly sketch the details of our column generation heuristic the LP relaxation.

We consider a master problem restricted to Inequalities (4.27), (4.4), and (4.43) and a very small set of initial paths $P \in \mathcal{P}_j$ for each demand j. We also add one tariff level $k \in K(e)$ for each edge e. We choose it as one that has minimum cost with respect to preprocessed flow bounds D_e (see Section 4.7). For all tariff levels and paths added, we also introduce the remaining of the Constraints (4.24), (4.28), (4.43) and (4.45).

We now address the pricing problem, which is stated as follows: Among the primal variables that are missing in the current formulation, find one with negative reduced costs, if it exists, to be added to the current formulation. In our case we distinguish two pricing problems for two different types of variables. The first one is to identify path variables r_j^P and amounts to a shortest path computation in the same network where the edge lengths are given by the dual variables associated with Inequalities (4.27). The second one is to find some tariff level variable y_k on an edge, that allows to lower the dual edge length variable.

The fact that both pricing problems affect each other makes pricing a difficult task. We have implemented and tested some preliminary heuristic that addresses the first pricing problem, while estimating the effect of the second pricing problem. In each round, it finds and adds a large number of paths and tariff level variables. Though meant as a proof of concept, it already yields promising results.

	evaluation criteria						
solution	nominal cost	historical average cost	worst-case-cost for Γ =2%	worst-case-cost for Γ =4%	worst-case-cost for Γ =8%	worst-case-cost for Γ =16%	worst-case-cost for Γ =32%
deterministic solution ($\Gamma = 0$)	**267,041**	**223,784**	578,039	659,397	734,394	807,057	912,221
robust solution for Γ =2%	278,880	225,292	**568,309**	**645,431**	719,675	794,308	899,490
robust solution for Γ =4%	287,056	226,422	573,620	646,783	717,552	792,874	895,072
robust solution for Γ =8%	294,385	230,483	580,477	649,090	**716,620**	**791,017**	893,848
robust solution for Γ =16%	307,593	237,456	592,023	659,590	721,882	792,087	894,053
robust solution for Γ =32%	311,567	240,519	594,669	662,491	724,539	792,684	**880,634**

Table 4.3: H_Auto_4 with 12 hubs and path and tariff level generation in 4 rounds

	evaluation criteria						
solution	nominal cost	historical average cost	worst-case-cost for Γ =2%	worst-case-cost for Γ =4%	worst-case-cost for Γ =8%	worst-case-cost for Γ =16%	worst-case-cost for Γ =32%
deterministic solution ($\Gamma = 0$)	**271,270**	**225,993**	585,821	664,346	742,326	814,526	920,067
robust solution for Γ =2%	284,427	227,210	**575,355**	**650,323**	726,186	799,460	903,657
robust solution for Γ =4%	568,567	288,371	851,796	924,150	995,792	1,068,505	1,169,544
robust solution for Γ =8%	302,698	232,758	587,760	657,608	**722,156**	**795,133**	**898,189**
robust solution for Γ =16%	587,396	297,319	866,790	935,392	996,661	1,066,486	1,162,412
robust solution for Γ =32%	594,216	301,952	874,198	940,765	998,808	1,067,626	1,155,905

Table 4.4: H_Auto_4 with 12 hubs and demand clustering into 4 clusters

4.9.3 Comparing clustering and generation

We compare both approaches on the very large scale test instance H_Auto_4. For the clustering we chose the number of partitions equal to 4 because this is the smallest number that allows acceptable computation times. For the same reason, the column generation method was limited to 4 rounds, but in each round we added up to one path per demand and one additional

tariff level per edge, if our heuristic found such a path/edge. Nevertheless a time limit of 18 hours was hit by the clustering approach three times and the respective solutions for $\Gamma = 4\%, 16\%$, and 32% had large gaps in the mixed integer programming solver. The average computation time was in both approaches similar and roughly 12 hours.

When comparing the column generation approach in Table 4.3 with the clustering techniques in Table 4.4, we first observe that the clustering solutions, that hit the time limit (for $\Gamma = 4\%, 16\%$ and 32%), can be considered a failure, because of unacceptable cost compared to those obtained by column generation. Also the two remaining clustering solutions are always dominated by the corresponding column generation solution, but the cost increase rarely exceeds 1%. A future research could also investigate how to combine these two approaches to solve even larger instances.

Similar to H_Auto_1, considerable savings in worst case cost can be achieved with robust solution, up to 3% e.g. for a robustness budget of $\Gamma = 32\%$ with the column generation approach. The price of robustness is however higher, with e.g. 7% for the same solution, when comparing the historical average cost.

4.10 Evaluation with Training and Test Sets

In this section, we test how our robust solution approach could perform under real demand data. Therefore we split the historical data into two parts, the first part is considered a training set and is used to generate uncertainty intervals for the demands. The second part is a testing set and we evaluate all solutions also with respect to the average observed cost, as well as the worst observed cost for a single week in the testing set. The splitting is done randomly 8 times which contributes 8 experiments for each case study. For H_Auto_1 16 out of 27 and for H_Auto_4 8 out of 17 observed weeks are drawn for the training set respectively. If a very sporadic demand does not occur in any scenario of the training set, we fix the lower and upper bound of its uncertainty interval to some small $\epsilon > 0$, which ensures, that a path is specified for this demand in a solution. We apply the path and tariff generation heuristic from Subsection 4.9.2 to obtain solutions for all uncertainty budgets as before. Now, for each experiment with a fixed training and testing set, we compute for each evaluation criterion (column) the percentage deviation from the best value that we observed for this criterion in the current experiment. Tables 4.5 and 4.6 report this deviation averaged over all 8 experiments.

Table 4.5 reports results for H_Auto_1. We observe only small differences in the average performance for observed scenarios (7th column), best performance is attained by robust solutions with uncertainty budget $\Gamma = 4\%$. These solutions also perform best for maximally observed scenarios (last column). For the solution specific worst cases (columns 2 to 6), robust solutions achieve average savings of more than 1% compared to the deterministic solution, even with moderate robustness budgets. For this instance robust solutions e.g. with budget $\Gamma = 4\%$ dominate deterministic ones on average.

However, for H_Auto_4 a higher price of robustness can be observed in Table 4.6: Good performance in the solution specific worst cases comes with moderate performance on the observed scenarios in the testing set. However the solution for $\Gamma = 2\%$ has a good tradeoff: More than 1.3% savings in worst case costs may outweigh a 0.05% cost increase for the observed scenarios on average. We also mention here, that H_Auto_4 is different from

	evaluation with intervals from training set					testing set evaluation	
solution	worst-case-cost for Γ =2%	worst-case-cost for Γ =4%	worst-case-cost for Γ =8%	worst-case-cost for Γ =16%	worst-case-cost for Γ =32%	avg. obs. cost	max. obs. cost
deterministic solution ($\Gamma = 0$)	0.89%	0.96%	1.15%	1.35%	1.73%	0.20%	0.83%
robust solution for Γ =2%	0.15%	0.18%	0.33%	0.48%	0.88%	0.05%	0.30%
robust solution for Γ =4%	0.14%	0.01%	0.13%	0.29%	0.71%	0.04%	0.18%
robust solution for Γ =8%	1.03%	0.63%	0.19%	0.21%	0.48%	0.36%	0.45%
robust solution for Γ =16%	1.35%	0.91%	0.16%	0.03%	0.13%	0.54%	0.50%
robust solution for Γ =32%	2.20%	1.64%	0.79%	0.44%	0.14%	0.66%	0.31%

Table 4.5: Training and testing set evaluation for H_Auto_1

	evaluation with intervals from training set					testing set evaluation	
solution	worst-case-cost for Γ =2%	worst-case-cost for Γ =4%	worst-case-cost for Γ =8%	worst-case-cost for Γ =16%	worst-case-cost for Γ =32%	avg. obs. cost	max. obs. cost
deterministic solution ($\Gamma = 0$)	1.35%	1.53%	1.91%	2.06%	2.96%	0.16%	0.28%
robust solution for Γ =2%	0.00%	0.10%	0.51%	0.65%	1.54%	0.21%	0.38%
robust solution for Γ =4%	0.70%	0.13%	0.11%	0.24%	1.10%	1.11%	0.98%
robust solution for Γ =8%	1.71%	0.63%	0.13%	0.10%	0.85%	2.25%	1.80%
robust solution for Γ =16%	2.11%	0.95%	0.23%	0.11%	0.65%	3.11%	3.06%
robust solution for Γ =32%	3.34%	2.13%	1.08%	0.64%	0.00%	4.74%	4.25%

Table 4.6: Training and testing set evaluation for H_Auto_4

H_Auto_1 in that it contains more sporadic demands. This has the effect that the nominal cost—that is the deterministic cost for demand values equal to the average observed—is considerably higher then the historical average cost, also compare Table 4.3. This might be a reason for the deterministic solution to perform best for observed scenarios in the testing set and we believe this could be fixed with a different method of interval generation, which is however beyond the scope of this work.

4.11 Conclusion

In this chapter we model strategic planning for routing multiple demands in a logistic network from a customer's perspective. The key features are the economies of scale and the consolidation of goods with complementary properties. Moreover, strategic planning has to

cope with uncertain demands.

We give an exact formulation for the robust counterpart of the resulting routing problem. As in this model already the adversary problem is NP-hard, we simplify the cost function and the adversary model to obtain a model that can be solved on real instances but preserves enough of the key features to produce competitive solutions. In fact, the resulting model shows structural similarities to facility location, which we see as an explanation, why the LP relaxation of our MILP approach turns out to be almost integral for the tested instances.

We test our method in a large real-world instance. To evaluated the robust cost of the routings found in this computational study, we can use the exact, NP-hard robust model. We also explore how the method can be extended to even larger instances by a column generation heuristics and a heuristic to partition the instances.

The results show that robust solutions are suboptimal for nominal demands, but may still perform well for the historical average. On the other hand they improve the worst-case cost by single digit percentages. This is independent of the precise level of robustness chosen for the optimization. The findings suggest that there are two kinds of historical good routings: those that are also robust and those that are not robust. Therefore, it seems worthwhile in logistic practice to pursue robust optimization in strategic planning.

Chapter 5

Hub Location for Logistics Networks

5.1 Introduction

In this chapter, we seek strategic routing as well as hub location decisions that determine future network operations, usually one year ahead. Hubs are special transshipment nodes of the network that allow for consolidation of flows. This helps to exploit economies of scale in the transportation tariffs compared to serving each origin–destination pair directly. In this chapter, we focus on networks where storage at hub nodes and exact timing of transports can safely be ignored or deferred to tactical or operational planning phases, for example an inbound logistics network from the automotive industry.

In practical applications, the location of hubs is typically restricted. Running a network with many hub nodes adds overhead to its management level. Thus, a compromise between low overall operational costs and a small hub number is sought. This gives rise to two kinds of optimization problems. The first are k-median variants, where a hard constraint limits the total number of hubs. The second are facility location variants where costs for opening hubs are incurred that may depend on the location. We focus on the first case, but also mention that our algorithms can treat the second case or a combination of both cases as well.

The costs of a planned hub configuration are determined by an underlying multicommodity flow problem. Each commodity is a triplet of origin, destination, and good. We call such a triplet a *demand*. The flow has no practically relevant capacity restriction but is limited to use the located hubs only. Moreover, the commodities have multiple attributes, such as mass, weight or volume. They determine the transportation cost on transportation relations of the network.

Transportation costs are governed by a market of freight forwarders. We use an accurate model of market tariffs that features discrete economies of scale. We distinguish less-than-truckload tariffs with up to 27 tariff levels, which are typically applied on spokes of the network, from full-truckload tariffs with one or two levels, which are typically applied on inter hub connections. The tariff levels can be understood in analogy to physical containers: the price of a container is constant no matter to which percentage it is filled. Hence the cost function has discrete jump points. Furthermore, higher tariff levels correspond to larger containers that come at a lower cost per size. Hence there are economies of scale. To assess the capacity of a container one has to evaluate two (or more) properties of the shipment, e.g., the maximal weight and the maximal volume. The size required for a shipment is the

maximum required size over all properties. This last feature of the cost function makes it advantageous to consolidate routes of heavy and light-weight goods.

5.1.1 Problem Description

We use the model for strategic route planning from Chapter 4 to deduce a model incorporating hub decisions.

However, here we use a smoothed notion of the transportation cost $c_e^T(a)$, compare with (4.1). It depends on the transported properties a and is defined as:

$$c_e^T(a) := \min_{k \in K(e)} (g_k \min\{m \geq 1, m \in \mathbb{R} : m\beta_{\pi k} \geq a_\pi \ \forall \pi \in \Pi\}) \tag{5.1}$$

According to the suggestion of our project partner 4flow AG [4fl17], this smoothing is better suited for the hub planning phase, as it better reflects the costs that are subject to adjustments during the tactical and operational planning phases. Such adjustments still have to fulfill service level requirements. Thus, it is important to keep the initial jump point of the cost functions and we require $m \geq 1$ in (5.1).

From the routing problem ROUTE from Section 4.4 with smoothed cost functions c_e^T, we deduce a corresponding hub location problem M-HUBROUTE. It is parametrized by a bound M for the maximum number of hubs to use: First, for a given subset $\bar{N} \subseteq N$ we define ROUTE($\bar{N}$) as the version of ROUTE (4.2)–(4.5) that restricts to hub nodes in $\bar{N}$. This means that all sets $\mathcal{P}_j$ are replaced by sets $\mathcal{P}_j(\bar{N})$ of source$_j$–sink$_j$ paths that only have hub nodes from $\bar{N}$ as inner nodes. We define the *M-hub location problem* as

$$M\text{-HUBROUTE} := \min_{\bar{N} \subseteq N, |\bar{N}| = M} \text{ROUTE}(\bar{N}). \tag{5.2}$$

A feasible solution $(\bar{N}, \mathcal{R})$ to M-HUBROUTE comprises a hub set $\bar{N}$ and a path solution $\mathcal{R} = (r_j^P)_{P \in \mathcal{P}, j \in J}$, that is feasible for the unrestricted problem ROUTE as well as for ROUTE($\bar{N}$). Apart from transportation costs, also opening costs may be incurred for each hub node that is used in a final solution.

Robust Counterpart

First observe, that a solution to the deterministic problem ROUTE is fully specified by a path solution $\mathcal{R} := (r_j^P)_{P \in \mathcal{P}, j \in J} \in \{0,1\}^{\mathcal{P} \times J}$ because all other variables and corresponding transportation costs can be deduced for deterministic and given demand values $d_j \in \mathbb{R}_+$.

We follow the methodology of [BS03] and define a robust counterpart in which d_j is uncertain in the interval $[\bar{d}_j, \bar{d}_j + \tilde{d}_j]$, the lower bound being the nominal value. Note, that we can exclude deviations below the nominal value for a worst case approach, as transportation costs are non-decreasing in demand values and thus such deviations will not worsen the costs.

For such intervals we consider classical Γ-restricted scenario sets, that is, we restrict by $\Gamma \in \mathbb{N}$ the number of demands deviating from their nominal value in a scenario. This yields the following scenario sets:

$$\{(d_j)_{j \in J} : d_j \in [\bar{d}_j, \bar{d}_j + \tilde{d}_j] \wedge \mid d_j \neq \bar{d}_j \mid \leq \Gamma\} \tag{5.3}$$

We obtain an adversary problem, denoted by ADV($\mathcal{R}$), that depends on a fixed path solution $\mathcal{R}$ and its induced set $E(\mathcal{R})$ of used edges. It is given by:

$$\text{ADV}(\mathcal{R}) = \max_{\mu\in\{0,1\}^J:\sum_j \mu_j\le\Gamma} \sum_{e\in E(\mathcal{R})} c_e^T\Bigg(\sum_{\substack{j:\exists P\in\mathcal{P}_j:\\ e\in P,\, r_j^P=1}} (\bar{d}_j+\mu_j\tilde{d}_j)\alpha_{j\pi}\Bigg) + \sum_{j\in J}\sum_{P\in\mathcal{P}_j}(c_j^P+\tilde{c}_j^P\mu_j)r_j^P \quad (5.4)$$

We can follow the steps in Section 4.4 to model ADV($\mathcal{R}$) as a mixed integer linear program (MILP) for fixed $\mathcal{R}$. The only difference is here that we have to account for the smoothing of c_e^T. However we can use variables $f_{k\pi}$ with the domain $f_{k\pi}\in\{0\}\cup[1,\infty]$ to model tariff level multiplicity. Such domains can be modeled in a MILP by introducing artificial binary variables.

This defines a robust counterpart ROBROUTE for ROUTE:

$$\text{ROBROUTE} = \min_{\mathcal{R}=(r_j^P)_{P\in\mathcal{P},\,j\in J}} \text{ADV}(\mathcal{R}) \quad (5.5)$$

$$\text{s.t.} \quad \sum_{P\in\mathcal{P}_j} r_j^P \ge 1 \qquad \forall\, j\in J, \quad (4.4\text{ rev})$$

$$r_j^P\in\{0,1\} \qquad \forall\, j\in J, P\in\mathcal{P}. \quad (5.6)$$

Again, this is not a MILP because ADV($\mathcal{R}$) is the optimum of an optimization problem. In the setting for hub location we also obtain a robust counterpart M-HUBROBROUTE for M-HUBROUTE:

$$M\text{-HUBROBROUTE} := \min_{\bar{N}\subseteq N,|\bar{N}|=M} \text{ROBROUTE}(\bar{N}). \quad (5.7)$$

5.1.2 Related Work

The problem of strategic planning of logistics networks integrates aspects of several classical optimization problems found in the literature. When ignoring hub decisions and restricting to one attribute of demands only, economies of scale on transportation relations typically give rise to concave cost functions. Thus we consider the concave multicommodity flow problem a central aspect of our problem. It has been addressed in a seminal paper by [GP90] where complexity issues, applications and solution techniques are discussed.

The case of concave piecewise linear functions can naturally be modeled as a fixed charge network flow. We refer to one classical and one more recent review by [MW84] and [Cra00], respectively. Recent progress on the state of the art of algorithms and solution techniques has been presented in [HNS10]. A success story is the application of Benders' decomposition. It allows to solve problems of moderate size to optimality. A recent survey can be found in [Cos05]. Note that the facility location version of hub restrictions can be considered a variant of network design, where hub nodes are replaced by edges that incur fixed costs corresponding to the opening cost for the hub.

When the number of hub nodes in solutions is restricted, or if using a hub incurs very high opening costs, then the resulting problems are considered the ones of hub location. In [AK08] and [CEK02] various models for hub locations are reviewed. All of them consider linear transportation costs on the network arcs where some discount factor α applies on inter hub arcs. Lately, [CML09] presented a hub location model with fixed opening costs at hub

nodes but no restriction on their number. It exhibits economies of scale for routing cost with three discount levels. They propose a Benders' decomposition that proves to be an effective approach for tackling instances up to 50 nodes.

When the number of hub nodes is constrained, the problem is referred to as the k-hub median problem. [AK08] report contributions where LP relaxations are found to be strong for some data sets. In many cases they deliver integral solutions immediately, or a branch-and-bound algorithm needs to explore only a small branch-and-bound tree.

In the terminology of [AK08], our model can be considered an extension of the k-hub median problem with multiple allocation to incorporate a tariff model for transportation cost for multi attribute demands. Multiple allocation allows to serve nodes where multiple demands originate to by several hubs.

In contrast to most of the models mentioned above, direct connections from origin to destination of a demand are always present in our case studies. We also note that many of the classical models for k-hub median problems reviewed in [AK08] give a path formulation for the underlying multicommodity flow, where paths are restricted to contain exactly two hub nodes. Our model has no such restrictions but handling costs at hub nodes typically limit the number of hub nodes occurring in relevant paths to satisfy the commodity demand.

We now address relevant literature to specific features of our models and algorithms.

Incremental Network Design When the networks show dynamics over time, hub decisions may be adapted to reflect changes of the network. This motivates the study to incremental solutions, where a chain of hub configurations is to be found, i.e. a strategy for opening hubs over time.

One of the first works on incremental network design is presented in [MP03] for the the k-median problem. A company has to build facilities, one at a time because of capital considerations, in order to supply its costumers. A solution is given by an order of the facilities to build. Its competitive ratio is the maximum ratio over k, of the assignment cost induced by opening the first k facilities in this order and the optimal cost of a k-median instance allowed to use all facilities. The authors present a hierarchically greedy strategy that leads to an algorithm with constant competitive ratio. [Lin+10] could improve this ratio and also extended the concept to various other cardinality constrained minimization problems, such as k-MST and k-vertex cover.

[KMS15] consider a setting in which arcs of a network can be added one by one in order to improve the value of a maximum s-t flow in the resulting network. The objective is to maximize the cumulative value of the maximum flows of each step. This is considerably different from a competitive ratio. They theoretically and empirically analyze the performance of mixed integer programming formulations and of natural heuristics.

[NS14] provide a more general classification of incremental network design problems. Here the change of the network is not limited to the creation of a single edge at a time, but is subject to a parallel machine scheduling environment. For various network performance metrics such as the shortest length of an s-t path, the minimum cost s-t flow value or the maximum s-t flow value, they show $\mathcal{NP}$-hardness. As a consequence, they study heuristics and present an algorithmic framework with a dispatching rule that can be applied to all of the incremental network design problems they study.

[AMS15] introduce a different notion of incremental network design for facility location. As in the model of [MP03], the task is to find an incremental sequence of facility sets to be

opened, but this time together with a corresponding incremental sequence of client sets to be served by these facilities. A competitive ratio is defined by comparing the incremental cost of the ℓ-th tuple of client and facility sets in the sequence against the cost of an optimum solution to an ℓ-robust facility location problem, which is a two-stage optimization problem.

Finally, [BDG17] propose a general theoretical framework to capture incremental solutions to cardinality constrained maximization problems. Their notion of *incremental problems* requires the cost functions to satisfy a monotonicity, sub-additivity and accountability property on the set of selected elements. They provide a general 2.618-competitive incremental algorithm for incremental problems, and show that no algorithm can have competitive ratio below 2.18 in general. They also analyze the exact competitive ratio of a natural greedy algorithm to be 2.313 in a setting when the cost function satisfies a relaxed submodularity condition.

Column and Row Generation One algorithmic aspect in our heuristics is simultaneous column-and-row generation, a technique that extends classical column generation, see for example [DDS05; LD05]. It has recently been used to solve large linear programs that otherwise hit memory restrictions. As one of the first contributions we are aware of, Avella, Sassano, and Vasil'ev [ASV07] managed to solve large scale k-median instances with up to 3795 nodes to optimality. They present a branch-and-cut-and-price solver that relies on a variant of column generation, which they call *delayed column-and-row* generation. The speciality lies in the absence of variables as well as rows in the restricted master problem.

Generic frameworks for column-and-row generation have lately been presented by Muter, Birbil, and Bülbül [MBB13], Sadykov and Vanderbeck [SV13], and Frangioni and Gendron [FG13]. The latter includes the application to multicommodity capacitated network design problem, and shows encouraging results. In our application, there are some structural similarities to the decomposition they use. For the pricing algorithms however, we have to pursue a different strategy. The frameworks of [MBB13; SV13; FG13] assume that the variables are grouped into *one* primal set and *one* secondary set, and that linking constraints between missing variables of both sets are omitted in a restricted master problem. Maher [Mah16] generalizes this to multiple sets of secondary variables and applies this to an integrated airline recovery problem. Nevertheless, this framework does not apply to our setting, as we have primary, secondary, and tertiary variables, and linking constraints between primary and secondary as well as secondary and tertiary variables are present.

5.1.3 Our contribution and outline of the chapter

We extend the detailed model in Chapter 4 to incorporate hub location decisions and obtain a k-hub-median model that exhibits detailed tariff modeling and multi-attribute demands. Because we face large scale instances in our case studies, we provide a broad algorithmic framework that contains LP-based as well as combinatorial heuristics for this model.

In Section 5.2 we develop solution techniques for the natural LP relaxation of the problem. We provide a novel application of column-and-row generation to logistics networks of medium sizes. The techniques differ from the frameworks of [Mah16] and [MBB13], as in our situation primary, secondary, and tertiary variables are present.

One difficulty for column-and-row generation is the absence of a dual solution: Since structural rows are missing in the primal restricted master problem the values of their corre-

sponding dual variables are unknown. The frameworks of [Mah16; MBB13] present efficient algorithms to compute such a dual solution by exploiting the special structure of primary and secondary variables. For our application, we rely on a restricted master problem for the dual linear program as well. In the pricing steps we construct a part of this dual master problem heuristically. Like this, the generation of primal variables is delayed until the dual restricted master problem has assembled enough information to price out a promising set of primal variables. We also establish an optimality criterion for this variant of column-and-row generation.

Section 5.3 focuses on incremental hub selection that aims to determine an incremental chain of hub sets. Selecting a hub set that belongs to an incremental chain increases the robustness of the selection in the context of growing or shrinking logistics networks. The presented heuristics feature different tradeoffs between solution quality and running time.

We also present two aggregation techniques that help to reduce the number of demands in a network. Using these techniques, even the slower heuristics for incremental network design can be applied to the largest networks of our case studies.

In Section 5.4 we evaluate the algorithms of Section 5.3 on seven networks provided by our industry partner 4flow AG [4fl17]. Using the LP-based search from Section 5.2, we also evaluate the price of being incremental by comparing hub configurations from an incremental chain against a best known non-incremental solution using the same number of hubs. To the best of our knowledge, this has not previously been done for instances from practice. Especially for the larger instances, the aggregation techniques are very important to obtain solutions. We also assess the loss in solution quality incurred by the aggregation: We apply our heuristics to the aggregated as well as to the original versions of the medium sized networks.

In Chapter 4, we focused on robust optimization and we extend this model. Thus, a natural question is whether our algorithms for hub location can be adapted for cost robust optimization under demand uncertainty as well. In Section 5.5 we affirm this for the LP-based search. Indeed, by heuristically anticipating dual variables for the adversary, we can use the deterministic techniques but lose the certificate for optimality typically inherent to column generation. Also, we derive uncertainty sets from historic data based on demand aggregation that differ from the ones in Chapter 4.

Finally, we also evaluate our algorithm for cost robust hub location on the two medium sized instances from the test set in Section 5.6. For this, we consider variants to derive uncertainty sets of historic demand information that aim at immunizing against anticipated trends or rough forecasts of changing demands.

5.2 LP-based Hub Search

The motivation in this section is to use the linearized model ROUTELIN from Section 4.6.1, to find heuristic solutions for the exact model ROUTE. The hope is that the error from using their linerized edge cost functions c_e^L instead of c_e^T on edges e is small enough such that optima of ROUTELIN and optima of ROUTE are close with respect to exact cost functions c_e^T.

We could observe that ROUTELIN tends to be strong for instances from practice, that is, its LP relaxation has optima that are close to optima of the MILP concerning their cost value. Moreover, it can be easily extended to incorporate hub decisions and a k-median constraint limiting the number of opened hubs by k. It is thus a natural first step to use solutions of the

corresponding LP relaxation as a starting point.

When applying the computational techniques from Section 4.7 to the LP relaxation of our instances we face new difficulties: First, for the test instances in Section 4.8 the hub location planning phase has been accomplished and the instances contain only a small number of selected hubs. The instances from this chapter have up to 100 potential hub nodes, which is a considerable blow-up in terms of required variables for possible transportation relations. Second, adding a M-median constraint makes the LP relaxation suffer from degeneracy: Applying classical column generation could often result in degenerate simplex iterations and grow the restricted master problem to an unmanageable size. Such an effect has also been observed by [ASV07] for large scale M-median problems.

To overcome these difficulties, we propose a column-and-row generation heuristic that we detail in the following. For the sake of a clearer presentation, we outline our algorithms only for the deterministic setting where we assume all demand values to be fixed. In Section 5.5 we discuss how to extend this to the cost robust setting as well.

Setting up the model HUBLINPATH We briefly outline the linearization steps from Subsection 4.6.1 to obtain a linearized model of ROUTE. Instead of exact cost functions c_e^T we use linearized tariff cost functions $c_e^L(d)$ for edges e that are defined by:

$$\ell_k : \mathbb{R}_+^J \to \mathbb{R}_+, \qquad \ell_k : d \mapsto \max_{\pi \in \Pi} \sum_{j \in J(e)} \sigma_{kj\pi} d_j \tag{4.17 rev}$$

$$c_k : \mathbb{R}_+ \to \mathbb{R}_+, \qquad c_k : \ell' \mapsto o_k + m_k \ell' \tag{4.18 rev}$$

$$c_e^L : \mathbb{R}_+^J \to \mathbb{R}_+, \qquad c_e^L(d) := \min_{k \in K(e)} c_k(\ell_k(d)) \tag{4.19 rev}$$

We used $\sigma_{kj\pi} := \alpha_{j\pi}/\beta_{\pi k}$ for shorter notation. The functions c_e^L determine in (4.19) the transportation costs as the minimum over tariff level costs c_k for tariff levels k from the set of available tariff levels $K(e)$ on edge e. Tariff level costs c_k are by (4.18) given as affine linear functions of the required fractional multiplicity ℓ_k to cover the properties of all transported demands with tariff level capacities, hence (4.17).

The linearized cost functions $c_e^L(d)$ can be expressed in a MILP as outlined in Subsection 4.6.1. We denote by HUBLINPATH a resulting formulation that we further extend to incorporate hub decisions: For each hub node n we introduce a binary variable g_n with cost coefficient o_n for opening this hub. The selection of at most M hubs is enforced by Inequality (5.10). We also give names to dual variables in brackets for each set of inequalities. The resulting MILP reads:

HUBLINPATH $=$

$$\min \quad \sum_{n \in N} o_n g_n + \sum_{k \in K} o_k y_k + \sum_{k \in K} m_k \ell_k + \sum_{j \in J} \sum_{P \in \mathcal{P}_j} c_j^P r_j^P \tag{5.8}$$

$$\text{s.t.} \quad \sum_{P \in \mathcal{P}_j:\, e \in P} r_j^P \le \sum_{k \in K(e)} z_{kj} \qquad \forall\, e, j \quad [RZ_{ej} \le 0]$$

(4.27 rev)

$$\sum_{j \in J} \sigma_{kj\pi} d_j z_{kj} + s_{k\pi} = \ell_k \qquad \forall\, k, \pi \quad [ZL_{k\pi} \in \mathbb{R}]$$

(4.28 rev)

$$\sum_{P \in \mathcal{P}_j} r_j^P \geq 1 \qquad \forall\, j \qquad [R_j \geq 0] \tag{4.4 rev}$$

$$y_k \geq z_{kj} \qquad \forall\, k, j \qquad [ZY_{kj} \geq 0] \tag{4.24 rev}$$

$$g_n \geq \sum_{k \in K(e)} y_k \qquad \forall\, n \in N,\ e \in \delta(n) \qquad [YG_{ne} \geq 0] \tag{5.9}$$

$$\sum_{n \in N} g_n \leq M \qquad [\kappa \leq 0] \tag{5.10}$$

$$r_j^P,\ z_{kj},\ y_k,\ g_n \in \{0,1\},\ s_{k\pi} \geq 0,\ \ell_k \in \mathbb{R} \tag{5.11}$$

General Outline of our Algorithms In the following sections, we present heuristic methods for solving HUBLINPATH. On the one hand, the hope is again that the integrality gap is small for instances from practice. On the other hand, already the LP relaxation is difficult to solve: Typically the number of demands is large and the sets of path variables $\mathcal{P}_j$ in HUBLINPATH grow exponentially in the input size of the networks, which prohibits applying standard solvers to the model.

From a practical perspective it may be feasible to restrict to paths with some maximum hop length. Typically, the handling costs at hub nodes make paths with more than 10 hub nodes too costly to be eligible in near optimum solutions. With such a restriction, the sets $\mathcal{P}_j$ are of polynomial size. This is still not enough to obtain tractable LP models and we choose not to enforce a maximum hop length explicitly. However, our heuristics are indeed tailored to use rather short paths in final solutions.

Another challenging aspect is the complex tariff structure. Note that introducing all variables z_{kj} and Inequalities (4.24) for all tariff levels k in $\cup_{e \in E} K(e)$ and demands $j \in J$ would result in more than a million inequalities and variables even for networks of reasonable size.

We propose heuristics that follow the paradigm of delayed column-and-row generation as presented in [Mah16], [MBB13]. The idea is to start with a very sparse set of variables and constraints of HUBLINPATH. Hence, we omit most of the possible paths in $\mathcal{P}_j$ and also most of the variables for tariff levels together with all associated linking variables and constraints. Omitted variables are assumed to have the value zero.

Observe that most of our inequalities have no constant part. Such inequalities are then feasible if none of the starting variables appear. The resulting formulation is called the current restricted master problem. Then, during the course of the algorithm we consider different pricing problems to identify sets of demands together with possible source–sink paths that have the potential to improve the current solution. Details are given in Section 5.2.1. The heuristics stop generating new paths after a specified number of rounds or some time limit. Clearly, it may have a fractional optimum. We turn it into an integral solution by applying some standard MILP software to this final formulation. Limiting the pricing rounds ensures that the final restricted master problem of HUBLINPATH remains amenable to branch and bound algorithms.

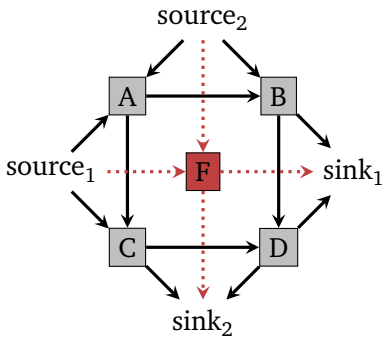


Figure 5.1: The network of Example 5.1 that shows an instance for which HUBLINPATH may have an unbounded integrality gap

Note that even if we were able to solve the complete LP relaxation of HUBLINPATH to optimality, a full-fledged branch and price algorithm would require to reinitiate column and row generation at each node in the branch and bound tree. We do not follow this strategy but merely use the MILP software here as a primal heuristic.

We close this section with a negative result concerning the integrality gap of HUBLINPATH from theoretical perspective, and supplement it with first computational aspects.

Example 5.1 (Unbounded Integrality Gap) *The integrality gap of HUBLINPATH may be unbounded. Consider the network depicted in Figure 5.1. Assume that demands j_1 and j_2 each need to send one unit of flow from their respective source node* source$_j$ *to the sink node* sink$_j$*. Two out of 5 hub nodes $A, \dots, F$ may be selected. Thick arcs represent low cost routing alternatives (say with a fixed cost of ϵ) whereas dotted arcs are costly alternatives (say of cost $M \gg \epsilon$ each). An optimal fractional LP solution is given by opening hubs $A, \dots, D$ at fraction 0.5 and splitting each demand equally onto the two disjoint source sink paths, avoiding hub node F completely. The cost of this solution is 12ϵ. However, any choice of two hubs out of $\{A, \dots, D\}$ does prohibit the routing of one of the two demands. Thus, the only feasible integer solution is to open hub F and to route all demands along paths over this hub, that is, to use dotted edges only. This integer solution thereby incurs a cost of $4M \gg 12\epsilon$. Now M and ϵ can be chosen to yield an arbitrary high integrality gap.*

5.2.1 Heuristics for Column Generation

We first discuss some computational aspects when trying to solve HUBLINPATH for our practical instances. First of all, Example 5.1 is rather artificial: In our instances there is always a possibility to route demands directly and avoid all hub nodes. Such solution with no hubs have for our instances roughly two times the cost of our best known solutions that are allowed to use all hubs.

Second, for larger instances with up to 3000 demands we can quickly reach memory limits when setting an initial restricted master problem of HUBLINPATH: We propose to add only 5-10 alternative paths for each demand plus the direct connection together with all variables for hub selection. Most of the tariff variables should be omitted. We refer to Subsection 5.2.7 for details on our method to choose a starting formulation.

If we use this method and solve the first restricted master problem of HUBLINPATH, we observe a major effect of Inequality (5.10): If M is large, say $\approx |N|/3$, then many of the hub decision variables g_n have already integral values. However, for more critical values of $M \in [4, 12]$ most of these variables have fractional values, i.e. they can be halfs, quarters, eights or an arbitrary fraction. This means that the current LP solution contains limited information on a good hub selection.

In the following we assume to have solved a restricted master problem for HUBLINPATH to optimality and that this solution is also feasible for the full problem. As the restricted master problem not only misses primal variables but also primal constraints, classical column generation method without row generation may insufficient, as discussed in [MBB13]. It can however still improve the current primal solution. We address two classical pricing problems related to finding missing tariff or path variables with negative reduced costs. We obtain them by looking at violated constraints of the dual of the restricted master problem of HUBLINPATH.

When we dualize the program of HUBLINPATH, we prefer to use nonnegative variables only. We replace $RZ_{ej} \leq 0$ with a variable $(-\widehat{RZ}_{ej}) \geq 0$ and $\kappa \leq 0$ with $(-\widehat{\kappa}) \geq 0$. We also replace the free variable $ZL_{k\pi}$ with $-\widehat{ZL}_{k\pi}$ for convenience. For the sake of readability however, we drop the hat notation, i.e. we write $ZL_{k\pi}$ instead of $\widehat{ZL}_{k\pi}$ again. Names for dual variables may reflect the primal variables that are coupled in the primal constraint corresponding to the dual variable. Let HUBLINPATHDUAL be the resulting dual program for HUBLINPATH.

HUBLINPATHDUAL $=$

$$\max \quad \sum_{j\in J} R_j - \kappa M \tag{5.12}$$

$$R_j \leq \sum_{e\in P} RZ_{ej} + c_j^P \qquad \forall j,\, P \in \mathcal{P}_j \qquad [r_j^P \geq 0] \tag{5.13}$$

$$RZ_{e(k)j} \leq \sum_{\pi\in\Pi} \sigma_{kj\pi} d_j ZL_{k\pi} + ZY_{kj} \qquad \forall k,\, j \qquad [z_{kj} \geq 0] \tag{5.14}$$

$$\sum_{j\in J} ZY_{kj} \leq \sum_{n\in N(k)} YG_{ne(k)} + o_k \qquad \forall k \qquad [y_k \geq 0] \tag{5.15}$$

$$\sum_{\pi\in\Pi} ZL_{k\pi} = m_k \qquad \forall k \qquad [\ell_k \in \mathbb{R}] \tag{5.16}$$

$$0 \leq ZL_{k\pi} \qquad \forall k,\, \pi \qquad [s_{k\pi} \geq 0] \tag{5.17}$$

$$\sum_{e:n\in\delta(e)} YG_{ne} \leq \kappa + o_n \qquad \forall n \in N \qquad [g_n \geq 0] \tag{5.18}$$

$$R_j,\, \kappa, RZ_{ej},\, ZY_{kj},\, YG_{ne} \geq 0,\, ZL_{k\pi} \in \mathbb{R} \tag{5.19}$$

We give a brief interpretation of HUBLINPATHDUAL as a charging scheme for overall transportation costs because this motivates the design of heuristics: We charge each demand with an individual price R_j and we try to maximize our profits with the sum of all demand prices in the objective function (5.12). On the other hand, we impose a global hub price κ that we pay in (5.12) for each of the M opened hubs.

The demand prices R_j are governed by prices for various resources and choosing a larger hub price κ allows for larger resource prices. Solving the dual is about optimizing this tradeoff. The resources are the hub nodes, edges and tariff levels. There is a demand specific cost

share RZ_{ej} for the usage of edge e, and an analogous cost share ZY_{kj} for the usage of tariff level k. For each tariff level there is also a price $ZL_{k\pi}$ for each property π. Finally, for each edge e, there are up to two edge prices $YG_{\text{start}(e)e}$ and $YG_{\text{end}(e)e}$ for incident hub nodes. These represent cost shares for the cost of hub resources, i.e. the global price κ together with a hub's opening costs o_n. Inequalities (5.18) link the edge prices YG_{ne} with the hub resource costs.

Similarly, the price R_j for some demand j is bounded in Inequalities (5.13) by the current cheapest cost of paths available for j in the current restricted master problem. The path cost is composed of the handling cost c_j^P and the individual edge prices RZ_{ej} for edges in the path P. Our incentive is to have the highest possible edge prices on edges that belong to shortest paths in order to maximize profits. However, RZ_{ej} is bounded by the costs for the available tariff levels on edge e in the right hand side of Inequalities (5.14). They are composed of the sum of all current property prices $ZL_{k\pi}$ multiplied with the demand's individual capacity usage $\sigma_{kj\pi}$ and a cost share ZY_{kj} for using this tariff level. By Inequalities (5.15) those shares compensate for the opening cost o_k of this tariff level and the edge price YG_{ne} of the underlying edge when accumulated over all demands.

As we use standard LP software to solve a current restricted master problem of HUBLINPATH, we can assume that we have a pair of feasible primal and dual solutions that are optimal for the current restricted master problems of HUBLINPATH and HUBLINPATHDUAL respectively. Due to the structure of the primal, we get that the primal solution is also feasible in the full formulation of HUBLINPATH but maybe not optimal. Of course, a necessary condition for optimality is that the dual solution is feasible for the full formulation of HUBLINPATHDUAL. So far "feasibility" for the full formulation of the dual is not well defined because it is not clear how to deal with the missing dual variables that are maybe required to have nonzero values.

In the light of classical pricing problems we can however check those missing dual constraints that do not contain any missing dual variables. This will yield the first two pricing problems. In the following, we also elaborate on a method to include missing dual variables and obtain a third pricing problem.

Pricing Problems

Problem 7 (PRICING PATH VARIABLES)

Given: Partial solution to a restricted master problem of HUBLINPATHDUAL with fixed variables $\overline{RZ}_{ej}$ for some subset of $E \times J$

Goal: Find a path $P \in \mathcal{P}_j$ which violates one of the Inequalities (5.13) for $\overline{RZ}_{ej}$.

Note that edge cost variables RZ_{ej} are only introduced on edges e that belong to at least one path alternative that is available for demand j. When restricting to already present edge cost variables we actually search for paths that are recombinations of edges from other available paths of this demand. This can be accomplished by shortest path computations with respect to current values $\overline{RZ}_{ej}$ as edge lengths.

Now we also would like to check the current dual solution against possible violations of inequalities of type (5.15) but we face one major problem: The primal Constraints (4.28)

and (4.24) are potentially missing from the restricted master problem of HUBLINPATH for some k, j, and so are the dual variables ZY_{kj} and $ZL_{k\pi}$. With simple observations we can however deduce the correct dual values which leads to pricing Problem 8 beeing polynomial solvable.

Problem 8 (PRICING TARIFF VARIABLES)
Given: Some edge e, fixed variables $\overline{RZ}_{ej}$ for some subset of J, fixed variables $\overline{YG}_{ne}$ for incident hubs.
Goal: Find a tariff level k for which Inequality (5.15) cannot be feasible.

Note that we still have $ZL_{k\pi}$ and ZY_{kj} as variables in Problem 8. We can rephrase it for each tariff level k as: min $\sum_j ZY_{kj}$ subject to (5.14) and (5.16) for the specific k with fixed values $\overline{RZ}_{ej}$ and $\overline{YG}_{ne}$. Inequality (5.15) for this k can be made feasible with these fixed values if and only if it is feasible for optimal values from the former optimization problem.

Observe that an optimum is attained when setting $ZL_{k\pi^*}$ to m_k for some π^* that has maximum $\sigma_{kj\pi^*}d_j$. This lets us compute all critical values $ZY_{kj} = \overline{RZ}_{ej} - m_k\sigma_{kj\pi^*}d_j$ and test each tariff level for violation of (5.15) in $O(|\Pi||J(e)|)$ time. Here $J(e)$ is the set of demands that are currently *assignable to edge e*, that is, there is some path alternative $P \in \mathcal{P}_j$ included in the current restricted master problem such that $e \in P$. If Problem 8 leads to a tariff level k to be added for some edge e, then we also introduce all corresponding variables z_{kj} for all demands j in $J(e)$.

The consequences of pricing tariff variables may be different depending on whether the underlying edge is active or inactive. An edge is called *active* if there is is at least one selected path in the current primal solution that contains this edge. On active edges it can directly improve the local transportation cost due to a cheaper tariff level being added. On inactive edges it most likely leads to degenerate simplex iterations without any improvement of the primal. It may however still reduce the dual cost shares ZY_{kj} of demands that are assignable at tariff level k.

In our algorithms we use Pricing Problems 7 and 8 with care, as they tend to spoil the primal restricted master problem with inequalities and variables that have limited impact on primal solutions but only tighten dual variables. On the other hand, tight dual information is helpful for finding variables to be priced out that lead to a non-degenerate simplex iteration.

The main motivation for our column-and-row generation heuristic is to delay pricing steps of Problems 7 and 8 and to aggregate their effect into a pricing LP. For this, we first turn the path formulation HUBLINPATH into an equivalent arc-node formulation HUBLINEDGE using classical network flow theory. We do not use HUBLINEDGE for our algorithms, but its dual is the basis for our pricing LP.

An arc-node formulation

In the LP-relaxation of HUBLINPATH, Inequalities (4.4) enforce the selection of one path for each demand j to send one unit of flow from each source$_j$ to sink$_j$. Yet we can enforce this flow also with inequalities of an arc-node formulation for the flow of demands and obtain the program HUBLINEDGE. Here, we require only weak flow conservation at nodes: Allowing

deficiency flow at nodes does not change the value of optimal solutions because additional flow is not beneficial when flow costs are assumed to be nonnegative.

Due to the special structure of our networks the Inequalities (5.23) for weak flow conservation occur only for hub nodes $n \in N$. Arbitrary deficiency flow may occur at sink nodes whereas at source nodes Inequalities (5.22) force one unit of outgoing flow. We also need to define the handling cost in terms of edges as $hand_j(e) := 1/2(hand_j(\text{start}(e))+hand_j(\text{end}(e))$. For the objective of HUBLINEDGE we account for the sum of path costs in HUBLINPATH now with a sum over edge handling costs.

HUBLINEDGE =

$$\min \quad \sum_{n\in N} o_n g_n + \sum_{k\in K} o_k y_k + \sum_{k\in K} m_k \ell_k + \sum_{j\in J}\sum_{e\in E} hand_j(e) x_j(e) \tag{5.20}$$

s.t. (4.28), (4.24), (5.9), (5.10)

$$x_j(e) \le \sum_{k\in K(e)} z_{kj} \qquad \forall\, e, j \qquad [RZ_{ej} \le 0] \tag{5.21}$$

$$\sum_{e\in\delta^+(\text{source}_j)} x_j(e) \ge 1 \qquad \forall j \qquad [R_j \ge 0] \tag{5.22}$$

$$\sum_{e\in\delta^+(n)} x_j(e) - \sum_{e\in\delta^-(n)} x_j(e) \ge 0 \qquad \forall j\ \forall n \in N \qquad [B_{jn} \ge 0] \tag{5.23}$$

$$x_j(e), z_{kj}, y_k, g_n,\ s_{k\pi} \ge 0,\ \ell_k \in \mathbb{R} \tag{5.24}$$

We now also give the dual HUBLINEDGEDUAL to HUBLINEDGE and also detail the correspondence of HUBLINEDGE and HUBLINPATH and their duals HUBLINPATHDUAL and HUBLINEDGEDUAL in the following paragraphs.

HUBLINEDGEDUAL =

$$\max \quad \sum_{j\in J} R_j - \kappa M \tag{5.12 rev.}$$

s.t. (5.14),(5.15), (5.16), (5.17), (5.18), (5.19)

$$R_j - B_{j\,\text{end}(e)} \le hand_j(e) + RZ_{ej} \quad \forall j,\ \forall e \in \delta^+(\text{source}_j) \quad [x_j(e) \ge 0] \tag{5.25}$$

$$B_{j\,\text{start}(e)} - B_{j\,\text{end}(e)} \le hand_j(e) + RZ_{ej} \quad \forall j,\ \forall e \in E \cap (N \times N) \quad [x_j(e) \ge 0] \tag{5.26}$$

$$B_{j\,\text{start}(e)} \le hand_j(e) + RZ_{ej} \quad \forall j,\ \forall e \in \delta^-(\text{sink}_j) \quad [x_j(e) \ge 0] \tag{5.27}$$

$$B_{jn} \ge 0 \tag{5.28}$$

It is a well known result from network flow theory that optimizing over the path formulation HUBLINPATH is equivalent to optimizing over its arc-node formulation HUBLINEDGE and that optimal solutions of HUBLINEDGE can be transformed into optimal solutions of HUBLINPATH and vice versa in polynomial time.

So for convenience we use the same names for dual variables in HUBLINEDGEDUAL and also in HUBLINPATHDUAL where possible: In particular, we chose the name for the dual variable R_j for Inequalities (5.22) in accordance with the dual variable for Inequalities (4.4) because they play the same role. We can interpret the dual values B_{jn} for some demand j

of optimal solutions as distance labels in a shortest path tree rooted at sink_j with respect to edge costs $c_j(e) := hand_j(e) + RZ_{ej}$. Then R_j is the special distance label at the source node. We also refer to B_{jn} as *node potentials*.

The equivalence of HUBLINPATH and HUBLINEDGE carries over to the restricted master problems when the restricted master problem of HUBLINEDGE contains exactly the same set of edges and flow variables x_j that are used by paths in the current restricted master problem of HUBLINPATH. Thus, asking for optimality of a primal solution in the restricted master problem of HUBLINPATH is the same as asking for optimality of a primal solution to the restricted master problem of HUBLINEDGE. Before we derive our pricing LP we need some general definition.

Definition 5.1 *For an LP in standard form $P = \{x \in \mathbb{R}^n, y \in \mathbb{R}^m \mid x, y \geq 0,\ A(x,y) = b\}$ such that the variables are partitioned into x and y, we call a partial solution $x \in \mathbb{R}^n$* feasibly extensible *in P if there exists $y \in \mathbb{R}^m$ such that $(x,y) \in P$.*

Note that this definition extends to general LPs as any LP can be expressed in standard form. We now turn to HUBLINEDGE and consider restricted master problems that correspond to restricted master problems of HUBLINPATH, i.e. feasible solutions with respect to variables $x_j(e)$ represent feasible multi commodity flows. Observe for such a restricted master problem of the primal program HUBLINEDGE that a feasible solution fulfilling Inequalities (5.22) for all demands $j \in J$ is also feasibly extensible in the full formulation of HUBLINEDGE: We can introduce all remaining variables and constraints and set their values to zero to obtain a feasible solution for the full formulation of HUBLINEDGE.

The situation is substantially different for a dual solution to a restricted master problem of HUBLINEDGEDUAL: Here, assuming zero for the values of missing variables is in most situations infeasible. Deciding optimality of the primal solution relates to feasibly extensibility of the dual solution. We state the following theorem:

Theorem 5.1 *Given a primal-dual pair $(\overline{S}^P, \overline{S}^D)$ of optimal solutions for current restricted master problems of HUBLINEDGE respectively HUBLINEDGEDUAL, where $\overline{S}^P$ satisfies every demand $j \in J$ with feasible Inequalities* (5.22)*. Further, let $\overline{S}^D$ be given by $(\overline{RZ}_{ej})_{j\in J, e\in\overline{E}(j)}$, $(\overline{ZY}_{kj})_{j\in J,k\in\overline{K}(j)}$, $(\overline{YG}_{ne})_{n\in N,e\in\overline{E}}$, $(\overline{ZL}_{k\pi})_{k\in\overline{K},\pi\in\Pi}$, $(\overline{B}_{jn})_{j\in J,n\in\overline{N}(j)}$, $\overline{\kappa}$ for suitable index sets $\overline{E}(j), \overline{K}(j), \overline{E}, \overline{K}, \overline{N}(j)$ that represent indices of constraints, that are already present in the current restricted master problem of the primal. Then we have that $\overline{S}^P$ is optimal for the full formulation of HUBLINEDGE if and only if $((\overline{RZ}_{ej})_{j\in J,e\in\overline{E}(j)}, (\overline{B}_{jn})_{j\in J,n\in\overline{N}(j)}, \overline{\kappa})$ is feasibly extensible in the full formulation of HUBLINEDGEDUAL.*

Proof. Suppose the solution $\overline{S}^P$ can be feasibly extended by values $(\widehat{RZ}_{ej})$, $(\widehat{ZY}_{kj})$, $(\widehat{YG}_{ne})$, $(\widehat{ZL}_{k\pi})$, $(\widehat{B}_{jn})$ for all the missing variables from the full formulation of HUBLINEDGEDUAL. This extension represents a feasible dual solution to HUBLINEDGEDUAL. Also the objective function value of this solution coincides with the objective function value of $\overline{S}^D$ since none of the extension variables has a nonzero coefficient in the objective function. If we also feasibly extend the primal solution $\overline{S}^P$ to the full formulation of HUBLINEDGE by setting all values for missing variables to zero, we obtain a primal-dual pair of optimal solutions for the full formulations.

Conversely, let $(\overline{RZ}_{ej})$, $(\overline{B}_{jn})$, $\overline{\kappa}$ be *not* feasibly extensible in the full formulation of HUBLINEDGEDUAL and consider a minimally infeasible subsystem that also contains the current

restricted master problem of HUBLINEDGEDUAL. Then there exists one inequality that we can remove to obtain a feasible subsystem. We call it a *critical* inequality.

A central observation is that a critical inequality can be chosen among inequalities of type (5.14): Suppose we remove all of these then we claim, that the remaining free variables can be adjusted to feasibility of remaining constraints. Consider first Inequalities of type (5.25-5.27) that are missing in the restricted master problem but are present in the minimal infeasible subsystem. Notice that for each demand j these inequalities correspond to the dual formulation of a shortest path problem with respect to variable edge costs RZ_{ej}. On their left hand side, the minimum path length $\overline{R}_j$ and possibly some node potentials $\overline{B}_{jn}$ may be fixed, but on their right hand side, the free edge cost variables RZ_{ej} can always be set arbitrarily high to guarantee that there is no shortcut path for j in the network, i.e. none of the new inequalities of type (5.25-5.27) becomes infeasible.

Similarly the variables in the left hand side of Inequalities (5.15), (5.18), and (5.16) can be adjusted to feasibility, as in the absence of (5.14), they are unresticted. Thus, we have moved any possible infeasibility into the yet missing Inequalities (5.14) as they link the variables RZ_{ej} with ZY_{kj} and $ZL_{k\pi}$.

With the removal of a critical inequality from (5.14) we obtain a subsystem of HUBLINEDGEDUAL in which the partial solution $(\overline{RZ}_{ej})$, $(\overline{B}_{jn})$, $\overline{\kappa}$ is feasibly extensible. Also, any such feasible extension does not increase the dual objective function value. We can now also augment the primal restricted master problem with all inequalities and variables that correspond to the variables and inequalities that we have augmented the dual with. Setting the values of new primal variables to zero yields a feasible primal solution for this augmented formulation, due to the special structure of HUBLINEDGE.

This way we obtain an augmented pair of optimal primal and dual solutions of the same objective value of $\overline{S}^P$. However, the critical dual inequality that we removed to attain feasibility is still violated which leaves a primal variable of negative reduced cost that is not yet present in the augmented primal solution. Thus, the primal solution $\overline{S}^P$ is not optimal. □

Now we can state the Pricing Problem 9 for simultaneous column and row generation by asking for feasibly extensibility of a dual solution that corresponds to a primal optimum in the current restricted master problem.

Problem 9 (PRICING PATH AND TARIFF VARIABLES)
Given: A current solution $\overline{S}^D$ to a restricted master problem for HUBLINEDGEDUAL.
Goal: Decide whether this solution is feasibly extensible to the full formulation of HUBLINEDGEDUAL or whether there is a set of variables and constraints such that their addition to current restricted master problem leads to violated inequalities in HUBLINEDGEDUAL and thereby prices out a primal variable.

Theorem 5.2 *Problem 9 can be solved in polynomial time.*

Proof. Even the full formulation of HUBLINEDGEDUAL has polynomial size and can be solved by the ellipsoid method in polynomial time, see [Kha80; GLS12]. We can fix all variables in the full formulation of HUBLINEDGE that are determined by $\overline{S}^D$ and still decide feasibility of the remaining problem in polynomial time. □

Problem 9 concerns the pricing problem for an arc-node formulation HUBLINEDGE but not for the original path formulation HUBLINPATH. Nonetheless, we will see in Subsection 5.2.2 that minimally infeasible subsystems of HUBLINEDGEDUAL, as described in the proof of Theorem (5.1), contain for each involved demand j a set of inequalities among (5.25-5.27). These represent shortcut paths between two nodes, for which the node potentials B_{jn} can be reduced. The shortcuts can be combined with paths that are already present in the restricted master problem of HUBLINPATH to price out new path variables. We will detail this approach in Subsection 5.2.2.

We also mention that the Pricing Problem 9 does not enforce the priced out variables to yield a non-degenerate simplex iteration. [MBB13] and [Mah16] propose to enforce this by adding suitable constraints to the pricing problem which might destroy structure of the pricing problem. Investigating whether these ideas can be practically applied to our problems remains for further research. However, we have some degree of freedom in choosing the variables to price out, i.e. which of the inequalities of HUBLINEDGEDUAL to check for infeasibility.

5.2.2 Solving Problem 9 practically: A violation LP

From preliminary tests restricted to pricing Problems 7 and 8, we could observe that although we could easily detect path and tariff variables with negative reduced costs, adding them to the formulation often causes degenerate simplex iterations only, that is, the values of dual variables may change but none of the primal variables change their value. Routing decisions are driven by hub decisions which in turn depend on the set of available routing alternatives for demands. Altering the current hub decisions requires to price out new path variables for many demands and possibly also some more tariff variables. The idea is to delay these pricing steps and to aggregate their impact in an LP that we call VIOLATION.

For a primal optimum of HUBLINPATH we compute a dual optimum $\overline{S}^D$ of the corresponding restricted master problem of HUBLINEDGEDUAL that contains the same set of edge assignments as HUBLINPATH. Note that $\overline{S}^D$ can be obtained using a dual optimum of the restricted master problem of HUBLINPATHDUAL and computing the node potentials $\overline{B}_{jn}$ as shortest path length from the each sink_j using edge costs $\overline{RZ}_{ej}$.

Instead of asking for feasibly extensibility of $\overline{S}^D$ in HUBLINEDGEDUAL we allow for violation of some constraints in the full formulation of HUBLINEDGEDUAL and set up a linear program with the objective to minimize the sum of violations of these constraints. Allowing for more flexibility, we let even some of the fixed values in $\overline{S}^D$ appear as variables again in VIOLATION.

We start with node potentials $\overline{B}_{jn}$ known from $\overline{S}^D$ that remain fixed and are the source of possible infeasibility. The idea is to capture a cascading structure of HUBLINEDGEDUAL that we detail next: Suppose one of the fixed distance labels $\overline{B}_{jn}$ is not correct. It will not lead to a violation of Inequalities (5.25) or (5.26) as long as there are edges e incident to n that allow for a too high value RZ_{ej}. These values RZ_{ej} in turn will not violate any inequalities of type (5.14) as long as the cost shares ZY_{kj} for corresponding tariff levels are still too high. And further, these cost shares do not lead to any violated Constraints (5.15) as long as (i) Not the suitable tariff levels k are introduced on edges. (ii) Not enough demands are assignable to these critical tariff levels (iii) Values of corresponding variables YG_{ne} are still too high. However, too high values of variables YG_{ne} on edges that are incident to some node n will finally lead to violated Constraints (5.18).

The main idea is now to accumulate the potential errors from too high values $\overline{B}_{jn}$ into a possible violation of Constraints (5.18) by using the non-fixed part of HUBLINEDGEDUAL for VIOLATION. We need the following notation: For some demand j let F_j be the set of nodes with $F_j \subseteq N \cup \{\text{source}_j, \text{sink}_j\}$, for which $\overline{B}_{jn}$ is fixed in VIOLATION. We obtain
VIOLATION $=$

$$\min \quad \sum_{n \in N} I_n \tag{5.29}$$

s.t. (5.14), (5.16), (5.17) as needed

$$\overline{B}_{j\,\text{start}(e)} - B_{j\,\text{end}(e)} \leq hand_j(e) + RZ_{ej} \quad \forall j,\ \forall e \in E \cap \big((F_j) \times (N \setminus F_j)\big) \qquad [x_j(e) \geq 0] \tag{5.30}$$

$$B_{j\,\text{start}(e)} - B_{j\,\text{end}(e)} \leq hand_j(e) + RZ_{ej} \quad \forall j,\ \forall e \in E \cap \big((N \setminus F_j) \times (N \setminus F_j)\big) \qquad [x_j(e) \geq 0] \tag{5.31}$$

$$B_{j\,\text{start}(e)} - \overline{B}_{j\,\text{end}(e)} \leq hand_j(e) + RZ_{ej} \quad \forall j,\ \forall e \in E \cap \big((N \setminus F_j) \times (F_j)\big) \qquad [x_j(e) \geq 0] \tag{5.32}$$

$$\sum_{e:n \in \delta(e)} YG_{ne} \leq\ \bar{\kappa} + o_n + I_n \qquad \forall n \quad [g_n \geq 0] \tag{5.33}$$

$$\sum_{j \in J} ZY_{kj} + \sum_{\pi \in \Pi} (ZL_{k\pi}/m_k) \sum_{j \in \bar{J}(k)} \overline{ZY}_{kj\pi} \leq\ o_k\ + \sum_{n \in N(k)} YG_{ne(k)} \quad \forall k \quad [y_k \geq 0] \tag{5.34}$$

$$\text{variable:} \qquad I_n, RZ_{ej},\ ZY_{kj},\ YG_{ne}, B_{jn} \geq 0, ZL_{k\pi} \in \mathbb{R} \tag{5.35}$$

$$\text{fixed:} \qquad \text{all } \bar{R}_j =: \overline{B}_{j\,\text{source}_j},\ \bar{\kappa},\ \text{and } \overline{B}_{jn} \text{ for } n \in F_j \tag{5.36}$$

VIOLATION has some modifications for inequalities from HUBLINEDGEDUAL that turned out to be useful: Inequalities (5.30-5.32) reflect the fixation of node potentials known from $\overline{S}^D$ and are thus a modified version of Inequalities (5.25-5.27). Some of the Inequalities (5.25), (5.26) or (5.27) with fixed values $\overline{B}_{jn}$ would have a modified counterpart of the type:

$$\overline{B}_{j\,\text{start}(e)} - \overline{B}_{j\,\text{end}(e)} \leq hand_j(e) + RZ_{ej} \quad \forall j,\ \forall e \in \delta^+(F_j) \cap \delta^-(F_j)\,. \tag{5.37}$$

We omitted such inequalities from VIOLATION because they imply fixed values of $\overline{RZ}_{ej}$ which can be preprocessed. These fixed values however can still contribute to some violation and are considered in relevant inequalities. Similarly if some property price values $\overline{ZL}_{k\pi}$ for some k are fixed from the current solution $\overline{S}^D$, they also imply fixed cost shares $\overline{ZY}_{kj}$ for those facilities. This time we would like to unfix them and allow for the flexibility of property prices $ZL_{k\pi}$ to be a variable in VIOLATION.

Therefore, we compute $\overline{ZY}_{kj\pi} := \max\{0, \overline{RZ}_{ej} - \sigma_{kj\pi} d_j\}$ to be the cost share if the property price $ZL_{k\pi}$ for π was set to m_k. We now modify Inequalities (5.15) to account for cost shares resulting from fixed values $\overline{RZ}_{ej}$ depending on the variable property prices $ZL_{k\pi}$. For some edge e, let $\bar{J}(e) \subseteq J$ denote the set of demands, for which variables $\overline{RZ}_{ej}$ are fixed. Similarly,

let $\bar{J}(k) \subseteq J$ be the corresponding set of demands for which the values $\overline{ZY}_{kj\pi}$ have been computed. Now for (5.15) we replace the sum of constants $\sum_{j\in\bar{J}} \overline{ZY}_{kj}$ of fixed cost shares with a variable term $\sum_{j\in\bar{J}(k)} \sum_{\pi\in\Pi} (ZL_{k\pi}/m_k)\overline{ZY}_{kj\pi}$ in the left hand side that accounts for the shares of $j \in \bar{J}(k)$ depending on the variable property price $ZL_{k\pi}$. We thus obtain Inequalities (5.34) in VIOLATION .

We also use Inequalities (5.14) and (5.16) from HUBLINEDGE and introduce all necessary variables. Finally, we allow for a violation of Inequalities (5.18) and capture this with additional variables I_n. Their sum is to be minimized in the objective function. The resulting Inequalities (5.33) also contain the fixed node toll $\bar{\kappa}$. Theorem 5.3 asserts that all modifications are in line with the notion of feasibly extensibility.

Theorem 5.3 *Given an optimum* $\bar{S}^D = \big((\overline{RZ}_{ej}), (\overline{ZY}_{kj}), (\overline{YG}_{ne}), (\overline{ZL}_{k\pi}), (\bar{B}_{jn}), \bar{\kappa}\big)$ *to a restricted master problem of HUBLINEDGEDUAL we have that* $\big((\overline{RZ}_{ej}), (\bar{B}_{jn}), \bar{\kappa}\big)$ *is feasibly extensible to the full formulation of HUBLINEDGEDUAL if and only if VIOLATION has an optimum of objective value zero.*

Before we give a proof of Theorem 5.3, we prove the following lemma, which is a direct consequence from our model for tariff cost.

Lemma 5.4 *W.l.o.g. we can assume for feasible solutions of HUBLINEDGEDUAL and VIOLATION that for each tariff level k there is at most one nonzero variable* $ZL_{k\pi}$ *equal to* m_k.

Proof. We show that we can turn any feasible solution of VIOLATION into a feasible solution with the desired property without changing the objective function value. Given such a feasible solution $(\widehat{I}_n), (\widehat{RZ}_{ej}), (\widehat{ZY}_{kj}), (\widehat{YG}_{ne}), (\widehat{ZL}_{k\pi}), (\widehat{B}_{jn})$ of VIOLATION, where for some fixed k^* more then two values among $\widehat{ZL}_{k^*\pi}$ are nonzero, then the all variables $\widehat{ZL}_{k^*\pi}$ and $\widehat{ZY}_{k^*j}$ contribute a feasible solution to the optimization problem

$$Q(k^*) := \min \quad \sum_{j\in J\setminus\bar{J}(k^*)} ZY_{k^*j} + \sum_{\pi\in\Pi} (ZL_{k^*\pi}/m_k^*) \sum_{j\in\bar{J}(k^*)} \overline{ZY}_{k^*j\pi} \tag{5.38}$$

$$\text{s.t.} \quad \widehat{RZ}_{e(k^*)j} \le \sum_{\pi\in\Pi} \sigma_{k^*j\pi} d_j ZL_{k\pi} + ZY_{kj} \qquad \forall j \in J \setminus \bar{J}(k^*) \tag{5.39}$$

$$\sum_{\pi\in\Pi} ZL_{k^*\pi} = m_k^* \tag{5.40}$$

$$ZY_{k^*} \in \mathbb{R}_+^{J\setminus\bar{J}(k^*)}, ZL_{k^*} \in \mathbb{R}_+^{\Pi} \tag{5.41}$$

in which all $\widehat{RZ}_{e(k^*)j}$ are fixed parameters and the objective function is derived from the left hand side of Inequalities (5.34). Altering all $\widehat{ZL}_{k^*\pi}$ and $\widehat{ZY}_{k^*j}$ to take the values of an optimum in $Q(k^*)$ does not affect the feasibility or the objective function value within HUBLINEDGEDUAL. It may however increase the slack of Inequalities (5.34). Further, we can assume that an optimum of $Q(k^*)$ attains equality in Inequalities (5.39) which allows us to substitute for

variables $RZ_{e(k^*)j}$ in $Q(k^*)$, and we obtain

$$Q(k^*) = \min \quad \sum_{j \in J \setminus \bar{J}(k^*)} \widehat{RZ}_{e(k^*)j} + \sum_{\pi \in \Pi} ZL_{k^*\pi} \left((1/m_k^*) \sum_{j \in \bar{J}(k^*)} \overline{ZY}_{k^*j\pi} - \sum_{j \in J \setminus \bar{J}(k^*)} \sigma_{k^*j\pi} d_j \right) \tag{5.42}$$

$$\text{s.t.} \quad \sum_{\pi \in \Pi} ZL_{k^*\pi} = m_k^* \tag{5.43}$$

$$ZL_{k^*} \in \mathbb{R}_+^{\Pi} \tag{5.44}$$

Obviously, an optimum of $Q(k^*)$ can be found, in which at most one of the variables $ZL_{k^*\pi}$ is nonzero.

It remains to show that also any feasible solution to HUBLINEDGEDUAL can be transformed into a feasible solution with the desired property. However, the argumentation for VIOLATION can be adapted to HUBLINEDGEDUAL in a straight forward manner. □

We are now able to prove Theorem 5.3.

Proof of Theorem 5.3. First, let $\hat{S}^D := \left((\widehat{RZ}_{ej}), (\widehat{ZY}_{kj}), (\widehat{YG}_{ne}), (\widehat{ZL}_{k\pi}), (\widehat{B}_{jn}) \right)$ be a feasible solution to VIOLATION with all values $\widehat{I}_n$ equal to zero. We claim that $\hat{S}^D$ leads to a feasible extension of $\left((\overline{RZ}_{ej}), (\overline{B}_{jn}), \bar{\kappa} \right)$ in the full formulation of HUBLINEDGEDUAL.

Observe that among constraints of type (5.16, 5.17, 5.25-5.27) each one is satisfied either by $\bar{S}^D$ for its version in the restricted master problem of HUBLINEDGEDUAL or by the composite solution $(\bar{S}^D, \hat{S}^D)$ for its version of the corresponding constraint of VIOLATION with

$$(\bar{S}^D, \hat{S}^D) := \left((\overline{RZ}_{ej}), (\widehat{RZ}_{ej}), (\overline{B}_{jn}), (\widehat{B}_{jn}), (\widehat{ZY}_{kj}), (\widehat{YG}_{ne}), (\widehat{ZL}_{k\pi}), \bar{\kappa} \right),$$

such that for each index combination there is at most one variable either from solution $\bar{S}^D$ or $\hat{S}^D$.

In some situations Inequalities (5.14) link solution $\bar{S}^D$ to $\hat{S}^D$, thus if (5.14) was feasible in HUBLINEDGEDUAL for $\bar{S}^D$ and some k, j, then this could no longer be the case for the composite solution $(\bar{S}^D, \hat{S}^D)$ because the right hand side needs to be adjusted to reflect the choice of $\widehat{ZL}_{k\pi}$ in VIOLATION. Observe for such inequalities that $\widehat{ZY}_{kj}$ is not present in VIOLATION as we use the preprocessed values $\overline{ZY}_{kj\pi}$ instead.

With Lemma 5.4 we can assume that there is π^* such that $\widehat{ZL}_{k\pi^*}$ equals m_k whereas $\widehat{ZL}_{k\pi} = 0$ for $\pi \neq \pi^*$ in VIOLATION. Observe that the setting $\widehat{ZY}_{kj} := \overline{ZY}_{kj\pi^*}$ leads together with $\widehat{ZL}_{k\pi}$ to an extension that fulfills (5.14) in HUBLINEDGEDUAL.

Also with such settings for all demands involved with some k we can substitute in the left hand side of (5.34) for $\widehat{ZL}_{k\pi}$ and $\overline{ZY}_{kj\pi}$ which yields

$$\sum_{j \in J} \widehat{ZY}_{kj} + \sum_{\pi \in \Pi} (\widehat{ZL}_{k\pi}/m_k) \sum_{j \in \bar{J}(k)} \overline{ZY}_{kj\pi} = \sum_{j \in J} \widehat{ZY}_{kj} + \sum_{j \in \bar{J}(k)} \widehat{ZY}_{kj} \leq o_k + \sum_{n \in N(k)} YG_{ne(k)} \tag{5.45}$$

and thereby shows that our extension also fulfills the corresponding Inequality (5.15) in HUBLINEDGEDUAL. Finally, Inequalities (5.18) are fulfilled by $(\widehat{YG}_{ne})$ and the fixed value $\bar{\kappa}$ because all the $\widehat{I}_n$ are zero in the corresponding Inequalities (5.33) in VIOLATION.

For the converse statement let $\hat{S}^D := \big((\widehat{RZ}_{ej}), (\widehat{ZY}_{kj}), (\widehat{YG}_{ne}), (\widehat{ZL}_{k\pi}), (\widehat{B}_{jn})\big)$ now be a feasible extension of $\big((\overline{RZ}_{ej}), (\overline{B}_{jn}), \bar{\kappa}\big)$ in HUBLINEDGEDUAL that has the property of Lemma 5.4. One can check that its restriction to the free variables in VIOLATION indeed constitutes an optimum with objective value zero. □

For practical applications we are not able to solve a full formulation of VIOLATION with standard LP-solvers with edge variables RZ_{ej} yet missing in the current restricted master problem. So in Subsection 5.2.3 we present heuristics that use VIOLATION to guide the search for small and promising sets of paths to be added to the current restricted master problem of HUBLINPATH in order to improve the primal solution.

5.2.3 Heuristic Search Strategies using VIOLATION

In order to motivate some sub-procedures of our heuristics we give an intuition for our case studies. Remember that all nodes in the network correspond to physical locations and although it is not part of the problem definition, cost functions for arcs of the network relate to physical distances. If we think locally for a single demand j, then we seek to reduce its cost share R_j, which is the sum of the edge cost shares RZ_{ej} along the currently selected path. Of course this cost share correlates with path distance but this correlation is limited. Think of a situation where there is spare capacity in some of the properties of an existing transport which leads to marginal transportation costs close to zero.

In near optimal solutions the selected paths for small demands tend to be detours when considering the distance information. These detours however enable the sharing of transportation cost with other demands and are shortest paths with respect to individual cost shares RZ_{ej}. Naturally however, there are limits on how much a detour can still be beneficial. The heuristic idea here is to use VIOLATION to search for path alternatives that are close to paths already present in the current restricted master problem. More precisely we restrict our heuristics to add variables and constraints to VIOLATION that offer one-hop alternatives to sub-paths of existing paths.

We present two major variants of our pricing heuristic that differ in the way they select possible one-hop alternatives. Their common part is listed as Algorithm 5.1. It can be subdivided into three phases. The call of a Procedure `hopHubs` in Line 2 makes the start of Phase I. The procedure carefully selects one-hop path alternatives and we give more details later. In Lines 4–6 the algorithm adds all necessary variables and constraints to VIOLATION for the new path alternatives: For example, if some new edge is introduced to VIOLATION, then we also need to add at least one tariff level, which is done in Lines 5–6. Here, we call a tariff level *active*, if it is present in the current restricted master problem and its variable has a positive value in the current solution.

The two variants of Algorithm 5.1 now differ in the method used for `hopHubs` in Line 2: There are the versions hopNewHubs and hopOpenHubs. Both search for candidate nodes n and demands j for which Inequalities (5.25) or (5.26) are missing from the restricted master problem on some incident edges. When they are added they hopefully generate a violation with respect to known distance labels $\overline{B}_{jn'}$ for nodes n' adjacent to n. In order to guide the search we introduce a *violation score* which is detailed in Subsection 5.2.4.

The main difference is the choice of hubs to include in one-hop alternatives. Here hopNewHubs realizes a broad search and considers all hubs but limits the number of demands

Algorithm 5.1: Outline of a path pricing step

Input: A restricted master problem currRMP of HUBLINPATH together with a current optimum and dual solution to the associated program HUBLINEDGEDUAL

Output: None, but for a set of demands $\tilde{J} \subseteq J$ we added paths $P_j \in \mathcal{P}_j$ for each $j \in \tilde{J}$ to currRMP that may generate violation w.r.t Inequalities (5.33)

1 Build the base version of VIOLATION with Inequalities (5.33) for all nodes in N ;
2 $E \times J \supset$ DemandEdgePairs := `hopHubs();` `// Phase I: one-hop alternatives`
3 **for** $(e, j) \in$ DemandEdgePairs **do**
4 Add Inequalities (5.30),(5.31) or (5.32) for (e, j) by creating the variable RZ_{ej} and also variables $B_{j\,\mathrm{start}(e)}, B_{j\,\mathrm{end}(e)}$ if needed;
5 Let k^* be the cheapest active tariff level on edge e, or the cheapest inactive level if none is active;
6 Assert presence of Inequalities (5.14) and (5.34) for level k^* (if not, add them) and add a new variable ZY_{k^*j} to the right hand side of (5.34);
7 Solve VIOLATION;
8 **for** $i = 1$ *to* $i = k_3$ **do** `// Phase II: add tight facilities`
9 **for** $e \in E$ **do**
10 **for** $k \in K(e)$ **do**
11 Let $J^a(k) := \bar{J}(k) \cup \{j \mid \exists n : (e, j) \in \mathrm{DemandEdgePairs}(n)\}$;
12 Compute $\overline{ZY}_{kj\pi}$ now for all $j \in J^a(k)$ and all $\pi \in \Pi$;
13 Let $\omega := \max_{k \in K(e)}\{\sum_{j \in J^a(k)} \sum_{\pi \in \Pi} (ZL_{k\pi}/m_k)\overline{ZY}_{kj\pi} - o_k - \sum_{n \in N(k)} YG_{ne(k)}\}$;
14 **if** $\omega > 0$ **then**
15 Add Inequality (5.34) and introduce all necessary variables for the tariff level k^* where the maximum ω is attained;
16 Solve VIOLATION;
17 PricedPaths :=selectPaths(VIOLATION) ; `// Phase III: Clean up`
18 **for** $(j, P) \in$ PricedPaths **do**
19 add P as path alternative for j in currRMP;

that can contribute to the violation at nodes. On the other hand hopOpenHubs first limits the set of considered hubs to include only hubs that are fractionally opened in the current optimum of the restricted master problem with a certain threshold. For those hubs all demands are considered that contribute any positive violation. The intuition for hopOpenHubs is twofold: First, it may help to reduce the number of hub decision variables with fractional value, and second, when a final LP formulation is passed to a branch-and-bound algorithm, it provides additional path alternatives that may become necessary when some of the hubs are forbidden in branching decisions.

In Phase II we consider all edges e in VIOLATION and check in Lines 10 to 14, whether some of the facilities $k \in K(e)$ might violate Inequality (5.34), and if so, we add it to VIOLATION. This step is very similar to the second pricing Problem 8 where we check against violation of Inequality (5.15) in the current restricted master problem of HUBLINPATHDUAL.

Given the greedy manner of Phase I, many of the added one-hop alternatives may be

superfluous. We want to limit the number of paths that we add in each pricing step. So in a clean up phase we greedily select a subset of paths alternatives trying to retain the violation at single hub nodes.

The main goal when designing Algorithm 5.1 is to maintain practical solvability of all included steps and its completion in reasonable running time. Since the search space of one-hop alternatives for all demands cannot be explored exhaustively, we have to restrict it by several means. Our principle here is greedy selection based on estimation: When we face many alternatives in some subroutine, we try to estimate the impact of each alternative with some score function and select a limited number of alternatives with highest score.

In order to control the width of the search, we introduce several parameters, where each one applies in a different situation. We list all such parameters in Table 5.1, together with a suggested setting that is tuned for our test cases. We now discuss the involved sub-routines.

5.2.4 Two Variants for Selecting One-hop Alternatives

Procedure demandCandidates(n)

Input: Current Primal solution to restricted master problem of HUBLINPATH and an associated dual solution to HUBLINEDGEDUAL, a hub node n

Output: a subset from $J \times \mathbb{R}_+$ that contains demands j paired with a violation score viol_j at hub n

1 **for** $j \in J$ **do**
2 $\mathsf{viol}_j := 0$;
3 **if** B_{jn} *is fixed* **then**
4 **for** $e := (u,n) \in \delta^-(n)$ *with fixed* B_{ju} *and* $j \notin \bar{J}(e)$ **do**
5 $\mathsf{viol}_j = \max\{\mathsf{viol}_j, \texttt{estLinEdgeCost}(e,j) - (B_{ju} - B_{jn})\}$
6 **for** $e := (n,v) \in \delta^+(n)$ *with fixed* B_{jv} *and* $j \notin \bar{J}(e)$ **do**
7 $\mathsf{viol}_j = \max\{\mathsf{viol}_j, \texttt{estLinEdgeCost}(e,j) - (B_{jn} - B_{jv})\}$
8 **else**
9 $\mathsf{maxSourceDist} := \max\{B_{ju} - \texttt{estLinEdgeCost}(e,j) \mid e = (u,n) \in \delta^-(n)$ and B_{ju} is fixed and $j \notin \bar{J}(e)\}$;
10 $\mathsf{minSinkDist} := \min\{B_{jv} + \texttt{estLinEdgeCost}(e,j) \mid e = (n,v) \in \delta^+(n)$ and B_{jv} is fixed and $j \notin \bar{J}(e)\}$;
11 **if** maxSourceDist *and* minSinkDist *exist* **then**
 $\mathsf{viol}_j := \mathsf{maxSourceDist} - \mathsf{minSinkDist}$;
12 **return** $\{(j, \mathsf{viol}_j) \mid j \in J$ and $\mathsf{viol}_j > 0\}$;

Both variants of Procedure `hopHubs` rely on a Procedure demandCandidates that selects a set of demands greedily based on a *violation score* viol_j for a given hub n which is formally defined later in Equation (5.47). We motivate this score as follows: Assume for some demand j a $\mathsf{source}_j \rightsquigarrow \mathsf{sink}_j$ path P is selected in the current solution and we consider some one-hop alternative u, v, w with $u, w \in P$ but $v \notin F_j$. If we assume the fixed node potentials $\overline{B}_{ju}$ and $\overline{B}_{jw}$ to be correct, then the edge costs $RZ_{(u,v)j}$ on (u,v) and $RZ_{(v,w)j}$ on (v,w) for j must satisfy:

$$RZ_{(u,v)j} + hand_j(u,v) + RZ_{(v,w)j} + hand_j(v,w) \geq \overline{B}_{ju} - \overline{B}_{jw} \tag{5.46}$$

This can be observed by adding up suitable inequalities from (5.25)–(5.27). To decide whether adding (u, v) and (v, w) to the formulation leads to a violation, we are interested in the parts of $RZ_{(u,v)j}$ and $RZ_{(v,w)j}$ that must be covered by ZY_{kj} for the tariff levels on (u, v) and (v, w) and which can then contribute to positive violation I_v. We first focus on $RZ_{(u,v)j}$ and its restricting Inequalities (5.14).

Recall from Equations (4.17) and (4.18) in Subsection 5.2 the definitions $\ell_k(d)$ for the maximum property usage for tariff level k with respect to a vector of demands d. We have for each tariff level k a fixed cost part o_k and a linear cost part $m_k\ell_k(d)$. The fixed cost could be arbitrarily shared by all demands involved in the usage of a tariff level on an edge whereas the linear part equals $\sigma_{kj\pi}d_j$ according to some selected property π on the tariff level.

To estimate the potential infeasibility of (5.46) we use an estimator function estLinEdgeCost: $E \times J \to \mathbb{R}_+$. It computes for an edge $e = (u, v)$ a pessimistic estimate for the linear cost part of $RZ_{(u,v)j}$. It then gives an optimistic estimate for the remaining fixed part of $RZ_{(u,v)j}$ that must then be attributed to the variables ZY_{kj}.

The function estLinEdgeCost anticipates a possible routing of demand j along edge e together with the current flow and selects the cheapest resulting tariff level k^* in Line 5 to compute a linear cost part $m_k\ell_{k^*}(d_j)$ for j as a cost estimate. Note that this estimate is not a lower bound on RZ_{ej} because the tariff level k^* is selected to be cost minimal for the joint flow on edge e. A true lower bound might be very close or equal to zero as there are tariff levels with a very high fixed cost part and a very low linear cost part. Using an estimate here helps to discard many of the demands with low violation score from consideration in demandCandidates.

Procedure estLinEdgeCost(e, j)

Input: Current primal solution $(\mathcal{R}, Z)$ to a restricted master problem of HUBLINPATH, some edge e and demand j

Output: Estimate for the linear cost part of RZ_{ej}

1 Set $d'(e) \in \mathbb{R}_+^J$ to the commodity flow on e according to $\mathcal{R}$;
2 **if** *j is not using e in $\mathcal{R}$* **then**
3 $\quad d'(e)_j := d_j$; `// assume that j will use e`
4 Let $k^* := \operatorname{argmin}_{k \in K(e)}\{o_k + m_k\ell_k(d'(e))\}$; `// k* is cheapest for d'(e)`
5 **return** $m_k\ell_{k^*}(d_j) + hand_j(e)$;

To define a violation score for $\mathsf{viol}_j(u, v, w)$ for the two edges (u, v) and (v, w) we replace the estimated parts in Inequalities (5.46) and obtain

$$\mathsf{viol}_j(u, v, w) := B_{ju} - B_{jw} - \texttt{estLinEdgeCost((u,v),j)} - \texttt{estLinEdgeCost((v,w),j)}. \quad (5.47)$$

We do not use $\mathsf{viol}_j(u, v, w)$ directly but rather score variants that consider minimization or maximization over suitable sets of edges. For example, Procedure demandCandidates computes for some hub n the maximum possible violation score viol_j for each demand over edges incident to n and returns a list of demand-score pairs. Depending on what variables in this inequality are already fixed in the current solution to the restricted master problem, the algorithmic steps are slightly different. Here the Lines 3–7 cover the case where the node potential B_{jn} is fixed and Lines 8–11 are for the case when B_{jn} is a variable in VIOLATION.

In all cases only hops u, v, w are considered, where the potentials of respective start and end nodes u and w are known.

Procedure hopNewHubs

Input: Current Primal solution to restricted master problem of HUBLINPATH and an associated dual solution to HUBLINEDGEDUAL

Output: A set of demand-edge pairs to that can be added to the current VIOLATION

```
1  candNodes := ∅;
2  for n ∈ N do
3      bestComms := {(j, viol_j) ∈ demandCandidates(n) with k_1 highest values viol_j};
4      nodeScore := Σ_{(j,viol_j)∈bestComms} viol_j;
5      candNodes := candNodes ∪ {(n, nodeScore, bestComms)};
6  for k_4 elements (n, nodeScore, bestComms) in candNodes with highest values nodeScore
   do
7      for (j, s) ∈ bestComms do
8          let Ẽ⁻ := {e = (u, n) ∈ δ⁻(n) | B_ju is known and j ∉ J̄(e)};
9          for k_2 elements e ∈ Ẽ⁻ with highest values B_ju − estLinEdgeCost(e, j) do
10             DemandEdgePairs := DemandEdgePairs ∪ {(e, j)};
11         let Ẽ⁺(n) := {e = (n, v) ∈ δ⁺(n) | B_jv is known and j ∉ J̄(e)};
12         for k_2 elements e ∈ Ẽ⁺(n) with lowest values B_jv + estLinEdgeCost(e, j) do
13             DemandEdgePairs := DemandEdgePairs ∪ {(e, j)};
14 return DemandEdgePairs;
```

Procedure hopNewHubs accumulates the demand-score pairs returned by demandCandidates into a violation score for hub nodes. We allow for k_1 demands with highest score to contribute to each hub node in Lines 2–5. This hub score is then used to restrict to k_4 hub nodes for which a preset number k_2 of incoming and outgoing edges are greedily selected in Lines 9 and 12 respectively.

Procedure hopOpenHubs selects one-hop alternatives to grow the program VIOLATION around hub nodes that are already fractionally selected in the current solution of the restricted master problem of HUBLINPATH, i.e. the value of g_n exceeds a threshold k_6.

To realize this, the loop in Line 2 iterates over all hubs n and demandCandidates is used to preselect $2k_1$ many demands that have a direct connection from their source node to n and a high violation score. We allow for roughly twice the number of demands compared to the setting in method hopNewHubs because the resulting program VIOLATION still solves rather quickly. For such a demand j, we first check whether we can add the direct hop via node n involving edges (source_j, n) and (n, sink_j) to VIOLATION in Lines 5 and 7.

In a second step, we would like to select additionally k_5 hub-hub edges for j that belong to one-hop path alternatives with a high violation score. Thus in Line 10 we iterate over all edges (n, v) in $\delta^+(n)$ and make sure in Lines 11 and 19, that either the node potential $B_{j,v}$ is known, or edge (v, sink_j) is present.

We compute a suitable violation score as before and select the k_5 edges with highest such score in Line 22. In this loop we also make sure that all edges for necessary one-hop paths are added to the returned list.

Procedure hopOpenHubs

Input: Current Primal solution to restricted master problem of HUBLINPATH and an associated dual solution to HUBLINEDGEDUAL

Output: A set $\{(e, j)\}$ of edge-demand pairs to add to the current VIOLATION

```
1  DemandEdgePairs := ∅;
2  for n ∈ N with g_n > k_6 do
3      bestComms := {(j, viol_j) ∈ demandCandidates(n) with 2k_1 highest values viol_j
         and (source_j, n) ∈ E};
4      for (j, s) ∈ bestComms do
5          if e := (source_j, n) ∈ E and j ∉ J̄(e) then
6              DemandEdgePairs = DemandEdgePairs ∪ {(e, j)};
7          if e := (n, sink_j) ∈ E and j ∉ J̄(e) then
8              DemandEdgePairs = DemandEdgePairs ∪ {(e, j)};
9          BestNeighbours := ∅;
10         for e := (n, v) ∈ δ⁺(n) with g_v > k_6 do
11             if neither B_jn nor B_jv is known then continue;        // only one-hops
12             if B_jn is known then
13                 sourceDist := B_jn;
14             else sourceDist := R_j − estLinEdgeCost((source_j, n), j) ;
15             if B_jv is known then
16                 sinkDist := B_jv;
17             else if (v, sink_j) ∈ E then
18                 sinkDist := estLinEdgeCost((v, sink_j), j);
19             else continue;
20             viol_j := sourceDist − sinkDist − estLinEdgeCost((n, v), j);
21             if viol_j > 0 then BestNeighbours = BestNeighbours ∪ {(e, viol_j)};
22         for k_5 elements ((u, v), viol_j) in BestNeighbours with best values viol_j do
23             DemandEdgePairs = DemandEdgePairs ∪ {((u, v), j)};
24             if (v, sink_j) ∈ E then
                 DemandEdgePairs = DemandEdgePairs ∪ {((v, sink_j), j)} ;
25 return DemandEdgePairs;
```

5.2.5 Extracting Priced Paths

If VIOLATION has a nonzero objective value then there exists a set of paths to be priced out and added to the current restricted master problem but we still need to identify the paths. Recall that VIOLATION is a slightly modified version of a dual linear program to an arc-node formulation of a multicommodity flow problem with special cost sharing structure. Thus the dual solution to VIOLATION represents a certain multicommodity flow, and we could identify the paths by giving a path decomposition of that flow. However, that flow will most likely use several and possibly disjoint paths for each demand, and adding them all could easily spoil our heuristic with too many variables.

We address the path extraction heuristically and take a greedy approach here. Our

goal is to focus on k_4 many hub nodes only and to select at most one path per demand for each selected hub, k_4 being a parameter of our heuristic. Now when looking at some hub n in a solution of VIOLATION, observe that the same demand j may contribute to I_n in Inequality (5.33) via variables YG_{ne} of *many* incident edges. Consider the modification of VIOLATION where we restrict a posteriori to one selected path per demand: In this version the values of YG_{ne} can only be reduced in optimal solutions and consequently also the values of I_n. The idea is now to look at each hub node n separately and to anticipate this reduction by selecting the demand paths greedily so as to retain a high value of I_n.

Recall VIOLATION reflects only one-hop alternatives for sub paths. For each demand and hub node we want to select one incoming and one outgoing arc. For some edge e that is incident to n, let $\hat{J}(e)$ be the set of demands with nonzero values ZY_{kj} in Inequality (5.34) for some $k \in K(e)$. Also, let $\hat{J}(n) := \bigcup_{e \in \delta^+(n) \cup \delta^-(n)} \hat{J}(e)$ be the set of demands involved in the violation at hub node n. So the task is to distribute the demands in $\hat{J}(n)$ once among the incoming edges of n and once among the outgoing edges.

Suppose we chose to assign some set $D \subseteq \hat{J}(n)$ to some outgoing edge $e = (n, v)$ and let $k(e)$ denote a tariff level $k \in K(e)$ for which Inequality (5.34) is tight in the current solution of VIOLATION, then for this edge we retain the following amount of violation:

$$\mathsf{viol}(D) := \left(\sum_{j \in D} ZY_{k(e)j} + \sum_{\pi \in \Pi} (ZL_{k(e)\pi} / m_{k(e)}) \sum_{j \in \overline{J}(k(e))} \overline{ZY}_{k(e)j\pi} - o_{k(e)} - YG_{ve} \right)^+ \tag{5.48}$$

For example if D is a set that includes all demands that contribute some positive $ZY_{k(e)j}$ to edge e, i.e. $ZY_{k(e)j} = 0$ for all $j \notin D$, and under a premise of Inequality (5.34) being tight, we could deduce $\mathsf{viol}(D) = YG_{ne}$ by resolving (5.34) for YG_{ne}. This means that we could retain all of the value YG_{ne}.

Procedure assignDemands greedily assigns demands $D \subseteq \hat{J}(n)$ to incident edges. It is called by Procedure selectPaths once for $\delta^+(n)$ and once for $\delta^-(n)$ and considers suitable sets $D = \hat{J}(e) \cap \hat{J}(n)$ that are edge and hub node dependent. It first selects some edge greedily that maximize the retained violation $\mathsf{viol}(D)$ in Line 2. The assignment is then stored as a mapping ϕ and the assigned demands are removed from any further considerations in Line 4. The process repeats until all demands from $\hat{J}(n)$ are assigned.

Procedure assignDemands

Input: A current solution of VIOLATION, some hub node n with demands $\hat{J}(n)$ contributing to violation I_n, some edge set $\overline{E}$ such that either $\overline{E} \subset \delta^+(n)$ or $\overline{E} \subset \delta^-(n)$

Output: A mapping $\phi : \hat{J}(n) \to \overline{E}$

1 **while** $\hat{J}(n) \neq \emptyset$ **do**
2 let $e^* := \operatorname{argmax}_{e=(u,v) \in \overline{E}} \left\{ \mathsf{viol}\left(\hat{J}(e) \cap \hat{J}(n)\right) \right\}$;
3 set $\phi(j) := e^*$ for all $j \in \hat{J}(e^*) \cap \hat{J}(n)$;
4 $\hat{J}(n) := \hat{J}(n) \setminus \hat{J}(e^*)$;
5 **return** ϕ;

Procedure selectPaths

Input: Current solution to VIOLATION

Output: A set PricedPaths $\subset J \times \mathcal{P}$ of demands j paired together with a path $P \subset \mathcal{P}_j$ to add to the current restricted master problem of HUBLINPATH

1 PricedPaths $:= \emptyset$, $\hat{\mathcal{P}}_j = \emptyset$ for $j \in J$;
2 **for** $n \in N$ **do**
3 $\quad \phi_+ :=$ `assignDemands` $\left(n, \hat{J}(n), \delta^+(n)\right)$, $\phi_- :=$ `assignDemands` $\left(n, \hat{J}(n), \delta^-(n)\right)$;
4 $\quad$ HubExtractViol$(n) := \sum_{e \in \delta^+(n)} \text{viol}(\phi_+^{-1}(e)) + \sum_{e \in \delta^-(n)} \text{viol}(\phi_-^{-1}(e))$;
5 **for** $n \in N$ *such that* HubExtractViol(n) *is among best* k_7 *values* **do**
6 $\quad$ compute ϕ_+ and ϕ_- as in Line 3;
7 $\quad$ **for** $j \in \hat{J}(n)$ **do**
8 $\quad\quad$ Let P be the path composed by:
$\quad\quad\quad$ source$_j \rightsquigarrow_j$ start$(\phi_-(j))$, n, end$(\phi_+(j)) \rightsquigarrow_j$ sink$_j$;
9 $\quad\quad$ PricedPaths $:=$ PricedPaths $\cup \{(j, P)\}$;
10 **return** PricedPaths;

Finally, Procedure selectPaths extracts a list of demand path pairs from VIOLATION to be added to the restricted master problem. In a preselection step in Lines 2 to 4, we sum the violation scores that assignDemands was able to retrieve at each hub node on incident edges and store it as HubExtractViol(n). We use this score to select k_7 hub nodes with highest such scores and combine the one-hop paths from assignDemands with paths in the current restricted master problem. To achieve this task we need the following observation: Consider the graph $\overline{G}_j$:

$$\overline{G}_j = (\{\text{source}_j, \text{sink}_j\} \cup N, E'(j)) \text{ with } E'(j) := \{e \in E \mid RZ_{ej} \text{ is known or implied}\}.$$

Also, for some demand j let $n \in F_j$ be a hub node where the potential B_{jn} is fixed in VIOLATION. Then $\overline{G}_j$ contains paths from source$_j$ to n and from n to sink$_j$. Let us denote with $s \rightsquigarrow_j t$ a shortest $s-t$-path in $\overline{G}_j$ with respect to $c_j(e) = hand_j(e) + RZ_{ej}$. Using this notion, we combine in selectPaths some one-hop path v, n, w for demand j from assignDemands with the paths source$_j \rightsquigarrow_j v$ and $w \rightsquigarrow_j$ sink$_j$. Their existence is guaranteed because we only consider one-hop paths v, n, w in VIOLATION for which v, w are in F_j.

5.2.6 Primal Heuristics

When inspecting an optimum of a current restricted master problem of HUBLINPATH, we typically find many hub decision variables g_n with fractional values. If we resolve the hub decisions, e.g. by rounding these variables in a certain manner to obtain a hub selection $\overline{N}$, then we are left to solve the routing problem ROUTE$(\overline{N})$.

The current restricted master problem of HUBLINPATH could be infeasible when restricted to the hub set $\overline{N}$ because of missing variables. Nevertheless, it might be not too difficult to recover a feasible solution. In Chapter 2 we consider similar routing problems that arise from a model for tactical transportation planning and feature a temporal pattern expansion. Among the combinatorial algorithms we presented a path based local search combined with

a starting solution from a suitable LP formulation. It yields strong results on the case studies therein. In the following we would like to adapt this local search to obtain feasible solution for a restricted master problem of HUBLINPATH quickly.

The path-based local search (LS) is based on a successive shortest path algorithm (SSP), Algorithm 2.2 from Subsection 2.4.2, that relies on a fast and heuristic resolution of a tariff selection sub-problem in order to compute the transportation cost for edges.

Since we use a different model for tariff cost in this work we have to address the resulting tariff selection sub-problem, which in our case is trivial: Given a vector $d(e) \in \mathbb{R}_+^J$ of demands using edge e in a current solution to HUBLINPATH we can obtain the transportation cost for edge e by

$$c_e^L(d) := \min_{k \in K(e)} c_k(\ell_k(d)). \tag{4.19 rev}$$

This computes very quickly as it amounts to $|K(e)|$ evaluations of corresponding functions c_k only. Another difference is that we work with path solutions, that is, we always maintain a path decomposition of the flow. Thus recomputing a path decomposition is not necessary in our case.

We also added two new features: First, the possibility to restrict the execution of SSP-steps to a subset of hubs $\overline{N} \subset N$, which means that all edges incident to some hub node n in $\overline{N}$ are discarded from consideration in the SSP-step. This leads to the second feature, the possibility to execute hub-rerouting steps: Given a set of hubs $\tilde{N} \subset N$ that we want to delete from a current solution, we delete all paths passing through at least one of the hubs in $\tilde{N}$ and perform SSP-steps restricted to $N \setminus \tilde{N}$. Note that in our model the presence of direct connections for each demand asserts that the last step is always feasible for any set $\tilde{N}$.

Combining both methods we denote with $\texttt{LS}(R, \overline{N})$ a local search step that transforms any solution R into one that uses none of the hubs $N \setminus \overline{N}$. In Subsection 2.6.3 we could observe that to reinitialize the starting flow pattern such that no direct connections are used can improve the result of a subsequent local search. We therefore also consider the variant $\texttt{REINIT}(\overline{N})$ that in a first step performs a SSP-step restricted to the network were all direct connections for demands are forbidden, and only paths via hubs in $\overline{N}$ are allowed. This step may leave some demands j unsatisfied in the situation when no source_j–sink_j path exists under these restrictions. It then completes this solution R with necessary direct connections for demands not yet routed, and finally performs a $\texttt{LS}(R, \overline{N})$ step.

5.2.7 Starting Formulation

We briefly outline how to obtain a starting formulation for the restricted master problem. The computational study in Section 4.8 suggests that one-hop path alternatives already reflect the main characteristics of the networks, so we use only a limited number of one-hop path alternatives in our starting formulation. The methods we propose base on the techniques from Section 4.7.

For each demand we choose k_8 paths with the following steps: First, we compute heuristic flow bounds $D(e) \in \mathbb{R}_+^J$ for each edge as outlined in 4.7.2. Assuming this bound was realized in a final flow pattern, we next compute *weighted volume cost share* $c_e^V(j, D(e))$ for each demand j, where $c_e^V(j, d)$ for a demand vector $d : J \to \mathbb{R}_+$ is defined as follows, also see (4.49): We fix an optimum tariff level k^* together with multiplicity f_{k^*} that is obtained when optimizing $c_e^T(a(d))$. Then, let the *property tightness* t_π for property π be defined as $t_\pi = a_\pi(d)/f_{k^*}\beta_{\pi k^*}$

and finally, let the *weighted volume cost share* $c_e^V(j,d)$ of demand j with respect to demand values d on edge e be defined as:

$$c_e^V(j,d) := g_{k^*} f_{k^*} \Big(\sum_{\pi \in \Pi} \alpha_{j\pi} d_j / f_{k^*} \beta_{\pi k^*} \Big) \Big/ \Big(\sum_{\pi \in \Pi} t_\pi \Big) \qquad \text{(4.49 rev)}$$

With the notion of cost shares $c_e^V(j, D(e))$ we select k_8 paths P with smallest scores $\sum_{e \in P} c_e^V(j, D(e))$ from the $|N|$ paths $P : \text{source}_j \to n \to \text{sink}_j$ with n in N as initial candidate paths for demand j.

After all paths are added we still have to add suitable tariff levels for edges. We know for each edge $e \in E$ a set $J(e)$ of demands that are assignable to edge e, that is, some path P with $e \in P$ has been added for j. Under the assumption that all demands in $J(e)$ would use the same tariff level on edge e we add one tariff level k^* for e that minimizes the simplified tariff cost $c_e^L(d(e))$ from (4.19), where $d_j(e) := d_j$ if and only if $j \in J(e)$ and otherwise $d_j(e) := 0$. With k^* we also add all necessary variables and constraints to make an assignment of demands in $J(e)$ to k^* possible in Constraints (4.28).

5.2.8 An LP-based Hub Location Heuristic

Procedure priceTariffLevels

Input: A restricted master problem currRMP of HUBLINPATH together with a current optimum and dual solution to the associated HUBLINPATHDUAL

Output: none, but a set of tariff levels with negative reduced costs are added to currRMP

```
1 for j = 1 to j = k_9 do                          // add facilities for edges
2     solve currRMP;
3     for e ∈ E do
4         ViolScore(e) = 0;
5         for k ∈ K do
6             solve Problem 8 and let v be the reduced cost for k;
7             update ViolScore(e) = min{ViolScore(e), v};
8     for at most k_10 edges e with lowest ViolScore(e) < 0 do
9         add Inequalities (5.15) and the required variables to currRMP;
```

Before we turn to our hub location heuristic, we first address the tariff pricing Problem 8 with Procedure priceTariffLevels: It performs k_9 consecutive rounds of tariff level pricing. The beginning of a round is the optimization of the current currRMP in in Line 2. Its optimum is the input for solving Problem 8 for each tariff level k in K in Line 6.

Naturally there will be many tariff levels with negative reduced costs and we restrict the number of simultaneously added ones by some number k_{10}. In each round, we select at most one level per edge and at most k_{10} tariff levels over all edges. The choice is realized by finding for each edge e one tariff level with most negative reduced cost ViolScore(e) in Line 7 and then, by selecting the k_{10} edges greedily in Line 8 for which their tariff level is added.

Algorithm 5.2: Outline of LP-based hub search

Input: An Instance of $M-$HUBROUTE
Output: A heuristic solution of the MILP HUBLINPATH for $M-$HUBROUTE

1 Build a starting formulation currRMP for an restricted master problem of HUBLINPATH;
  `// see Section 5.2.7`
2 **for** $i = 1$ *to* $i = k_{11}$ **do** `// pricing rounds`
3 priceTariffLevels(currRMP);
4 solve currRMP and store solution as LPSol ;
5 **if** $i \leq k_{12}$ **then** `// global phase`
6 **if** i *is even* **then** call Algorithm 5.1 for currRMP with hopNewHubs ;
7 **else** call Algorithm 5.1 for currRMP with hopOpenHubs ;
8 **if** $i > k_{12}$ **then** `// local phase`
9 $\overline{N} :=$ { the M hubs n with highest values g_n in LPSol };
10 $\mathsf{PathSol}_1 := \mathtt{LS}(\mathsf{LPSol}, \overline{N})$;
11 $\mathsf{PathSol}_2 := \mathtt{REINIT}(\overline{N})$;
12 **for** $j \in J$ **do**
13 **if** $P_j \in \mathcal{P}_j$ *is used in* $\mathsf{PathSol}_1$ *or* $\mathsf{PathSol}_2$ *but missing in currRMP* **then**
14 add P_j to currRMP;
15 turn currRMP into a mixed integer program currMILP by adding the required integrality constraints and solve it with a standard branch-and-bound algorithm;
16 **return** a near optimal solution of currMILP ;

We now present The LP-based hub search, listed as Algorithm 5.2, to select M hubs from the set N. After a starting formulation has been created as described in Subsection 5.2.7, the main loop in Line 2 performs pricing rounds for the current restricted master problem currRMP that each consist of a search for new tariff levels with negative reduced costs in Line 3 and different hub search heuristics for the remaining part.

After k_{11} of these major pricing rounds we turn the final currRMP into a mixed integer linear program currMILP by imposing integrality of all binary decision variables. It is then solved with a standard branch-and-bound algorithm. Note that the most influential binary variables in currMILP are hub decision variables g_n: In the branch where some g_n is forced to zero all path alternatives using hub n can not be selected anymore. It is thus a reasonable suggestion to give higher branching priority to those variables. For large instances it is necessary to enforce a global time limit as Step 15 is unlikely to terminate with proven optimality.

The pricing rounds in the loop of Line 2 can be attributed to a global phase for the first k_{12} rounds and a local phase for the remaining rounds. In the global phase in Line 5 to 7 we call the hub pricing Procedures hopNewHubs and hopOpenHubs alternatingly. The hope is that additional routing alternatives for hubs may produce LP optima that have lower cost and show a reduced number of fractional hub decision variables. In the local phase, we call the primal heuristics from Subsection 5.2.6 and restrict them to the M hubs that have the highest value g_n in the current solution. The idea is to fix the hub decisions that emerged from the global phase and to search for paths that have been missed in the heuristic hub

search pricing steps.

Further Algorithmic Details We close this section by giving details that we consider helpful for an implementation of the above algorithms. Note that after variables have been added to the restricted master problem, the update steps are referred to as "solve" statements. Of course a generic LP-solver typically performs effective warm start heuristics in such situations to recover the previous solution, which is important for the effectiveness of our algorithms.

More generally, most of our procedures rely on a broad information context, e.g. concerning the network data or temporary information of the environment they are called from. We do not make this information passing explicit, but confine ourselves to mentioning only objects that are helpful for the readability or that are altered during the execution of a procedure.

The many tuning parameters that occur in the subroutines are listed in Table 5.1. They control the size of the search spaces and are the result of many preliminary tests that we conducted for medium sized instances from our case studies.

Table 5.1: Tuning parameters of sub routines used in Algorithm 5.2

parameter	setting	explanation
k_1	$\lvert J\rvert/M$	how many demands to add at each hub node in Algorithm 5.1
k_2	8	how many outgoing respectively incoming edges to select per hub node, per demand in hopNewHubs
k_3	3	how many rounds of finding violated facilities in Algorithm 5.1
k_4	10	how many node neighbors to add in hopNewHubs
k_5	5	how many node neighbors to add for each demand and for each hub node in hopOpenHubs
k_7	10	threshold for selectPaths, for how many hubs to add paths
k_8	9	how many paths to add for each demand in a starting formulation, see Section 5.2.7
k_9	5	how many rounds of tariff level tightening in Line 1 of Procedure priceTariffLevels
k_{10}	25	how many facilities to add to Line 8 in one tightening round of Procedure priceTariffLevels
k_{11}	10	how many pricing rounds to perform in Line 2 of Algorithm 5.2
k_{12}	5	how many global pricing rounds to perform in Line 5 before switching to local rounds in Line 8 of Algorithm 5.2

5.3 Incremental Hub Chains

In Section 5.2 we considered an LP-based search for the $M-$HUBROUTE problem, where the hub bound M is part of the problem input. Practitioners designing a network seek to evaluate several solutions for different hub bounds and may as well take measures other than costs into account. Logistics networks may show long-term dynamics and hub decisions need to be adapted at a later point in time to reflect changes of the network. This means that additionally required hubs have to be opened or superfluous hubs have to be closed. To this end, we consider incremental solution chains: For each k in some interval $[L, M]$ we seek

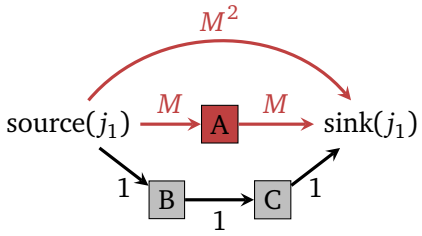


Figure 5.2: Graph for Example 5.2, edge labels refer to cost.

a hub set of cardinality at most k and an associated solution that uses only hubs from the set. Moreover, we require the hub sets to build a chain of inclusion. Implementing a solution from the chain makes the network more robust against changes to the network.

The notion of incremental hub chains raises the question of how hub selections that are part of a chain compare to unconstrained hub selections, or phrased differently: What is the price of being incremental?

The study of incremental solutions is also interesting from the algorithmic perspective: The methods of an LP-based hub search from Section 5.2 are limited to instances with medium sized sets of demands. In this section we develop a decremental algorithmic framework for incremental solutions that also contributes combinatorial heuristics which are competitive with an LP-based hub search when considering the problem M-HUBROUTE. Moreover, when combined with aggregation techniques, these algorithms are able to solve even the largest instances in our test set. We now formalize the above notions of incremental solutions and a price of being incremental.

Definition 5.2 (Incremental Solutions) *Given hub bounds $L < M$ for a logistics network, then a sequence of solutions $(\bar{N}_L, \mathcal{R}_L), \ldots, (\bar{N}_M, \mathcal{R}_M)$, such that $|\bar{N}_h| \leq h$ holds for each h, is called an* (L, M)-incremental solution*, if and only if $(\bar{N}_h, \mathcal{R}_h)$ is a solution to $h-$*HUBROUTE *and the hub sets form a chain $\bar{N}_L \subseteq \bar{N}_{L+1} \subseteq \ldots \subseteq \bar{N}_M \subseteq N$.*

Here the lower and upper bounds L respectively M are considered as problem input, e.g., a practitioner may have a clear understanding of some minimal or maximal number of hub nodes for the network. Note that the routings associated to different hub sets is allowed to completely independent. A first natural question is whether the h-part $(\bar{N}_h, \mathcal{R}_h)$ of a (L, M)-incremental solution has inherently higher cost than an optimum to $h-$HUBROUTE. In this regard we define a price of being incremental and show with Example 5.2 that it can be unbounded in general.

Definition 5.3 (Price of Being Incremental) *Given an incremental solution $(\bar{N}_L, \mathcal{R}_L)$, $\ldots$, $(\bar{N}_M, \mathcal{R}_M)$ for a logistics network, we define the* price of being incremental *for this solution to be*

$$\sup_{h=L,\ldots,M} \frac{cost\big((\bar{N}_h, \mathcal{R}_h)\big)}{cost\big(OPT(h)\big)}. \tag{5.49}$$

Example 5.2 (Unbounded Price of Being Incremental) *Consider the network depicted in Figure 5.2. Assume that demand j_1 needs to send one unit of flow from* source(j_1) *to* sink(j_1).

An incremental hub chain either selects both hubs B and C in any order, or leaves one of them unselected in favor for hub A. In both situations the price of being incremental equals M and can be set arbitrarily high.

5.3.1 Decremental Algorithms

We present with Algorithm 5.3 a decremental framework to compute incremental solutions that is natural and simple. We start by computing a routing solution that may use all possible hub nodes using some routing algorithm A. Then we successively close one hub after another to obtain routing solutions using again algorithm A for each of the hub bounds h, $L \leq h \leq M$.

An important ingredient is the weighting function $p : N \times \text{SOL} \to \mathbb{R}_+$ that computes a heuristic score for each hub that is open in a current solution. The score is used to guide the decision, which hub to close next. In the following we present the variants of Algorithm 5.3 that we tested and specify for each variant the function p and the Algorithm A.

Algorithm 5.3: `Decr-Greedy`(p,A) – framework for decremental hub selection

Input: A logistics network with hub set N of potential hub nodes, hub bounds L, M, a solution dependent weight function $p : N \times \text{SOL} \to \mathbb{R}_+$, a routing algorithm A to solve instances of ROUTE

Output: A (L, M)–incremental solution

```
1 N̄ := N;
2 for h = |N|, ..., L do
3     compute a solution (N̄_h, R_h) to ROUTE(N̄) using algorithm A;
4     let n_h = argmin_{n∈N̄} p(n, R_h) and set N̄ := N \ {n_h};
5 return (N̄_L, R_L), ..., (N̄_M, R_M)
```

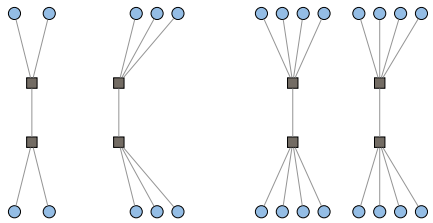
Heuristic Decrements A natural weight function p is to measure the usage of each hub node. A usage measure can be specified by computing the throughput of relevant properties $\pi \in \Pi$, or simply by counting the number of demands that are routed along each hub. For a path solution $\mathcal{R} = (r_j^P)_{P \in \mathcal{P}, j \in J}$ we define the *property usage*

$$\text{USAGE}_\pi(n, \mathcal{R}) := \sum_{j \in J} \sum_{P \in \mathcal{P} : n \in P} r_j^P d_j \alpha_{j\pi} \tag{5.50}$$

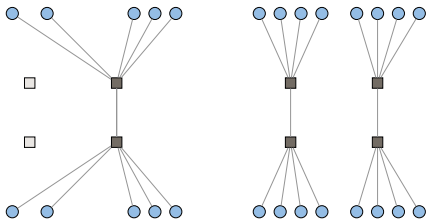
and the *demand usage*

$$\text{USAGE}_\#(n, \mathcal{R}) := |\{j \in J : \exists P \in \mathcal{P} \text{ with } n \in P \text{ and } r_j^P = 1\}|. \tag{5.51}$$

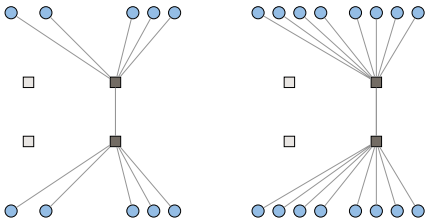
Iteratively closing hubs with some lowest value USAGE follows the intuition that a hub's usage correlates with its potential to save cost. The limitations of this intuition become obvious in Example 5.3 where $\text{USAGE}_\#(n, \mathcal{R})$ is used in Algorithm 5.3: It shows that closing hubs greedily with respect to some USAGE may introduce errors that lead to suboptimal solutions. We therefore try to keep these errors as small as possible by closing at most one hub at each iteration in Algorithm 5.3.



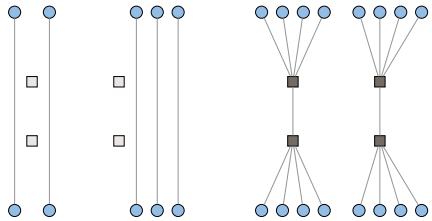
(a) Solution without hub limitation.



(b) Solution after first iteration of Algorithm 5.3.



(c) Solution after second iteration of Algorithm 5.3.



(d) Selection of the two mostly used hubs from (a).

Figure 5.3: Limitations of the greedy intuition: Comparison between keeping the mostly used hubs (d) and closing one hub at a time iteratively (c).

Example 5.3 (Step Size for Hub Closing) *In this example we use* $\text{USAGE}_{\#}(n, \mathcal{R})$ *to guide hub decisions. It shows that closing more than one hub at a time greedily may lead to a bad solution. It is depicted in Figure 5.3. Here closing the two least utilized hubs of a starting solution with no hub restrictions (a) leads to a solution where many direct connections are used (d). Since direct connections prohibit the consolidation of flows, we expect solution (d) to show significantly higher cost than solution (c) that was obtained by our decremental greedy frame work.*

An alternative with more computational overhead is to pre-compute the impact of closing each hub explicitly. We introduce a weight function $\Delta(n, \mathcal{R}_h)$ to use for p in Algorithm 5.3, Line 3, to be evaluated as follows: We tentatively close each hub $n \in \bar{N}$ and perform a re-routing step $\text{LS}(\mathcal{R}_h, \bar{N}_h \setminus \{n\})$. The function $\Delta(n, \mathcal{R}_h)$ is then defined as the change in cost when considering to close hub n:

$$\Delta(n, \mathcal{R}_h) := \text{cost}\left(\text{LS}(\mathcal{R}_h, \bar{N}_h \setminus \{n\})\right) - \text{cost}\left(\mathcal{R}_h\right) \tag{5.52}$$

The overhead of the re-routing step significantly slows down this variant of Algorithm 5.3. Indeed the resulting running time is unacceptable for our computational study so we investigate several speed-up techniques in Subsection 5.3.2 that come with a loss in solution quality. This however enables us to include this variant in our computational study.

Routing Algorithms In Subsection 5.2.6 we already adapted routing algorithms from Chapter 2 for the use as a primal heuristic in Algorithm 5.2. The heuristic REINIT($\bar{N}$) provides a very fast routing algorithm and is thus a natural candidate for A in Algorithm 5.3.

Moreover, we can also use the whole Algorithm 5.2 not only to obtain a solution to $M-$HUBROUTE but also as a routing algorithm for instances of ROUTE($\bar{N}$) for some hub set $\bar{N}$. Since REINIT($\bar{N}$) is a subroutine of Algorithm 5.2 we expect strictly better results for this choice of algorithm A. It is not applicable to very large instances because running time and required memory exceeds our computational possibilities, but it still allows a comparison on medium sized instances with other variants.

Implicit Decrements Another variant delegates the decision which hub to close next completely to Algorithm 5.2. This is listed as Algorithm 5.4: In the first iteration of the main loop, we compute a solution to $M-$HUBROUTE using Algorithm 5.2 in Line 3 and store the set $\bar{N}$ of used hubs. Then in subsequent iterations, we reapply Algorithm 5.2 to compute a solution to $(|\bar{N}|-1)-$HUBROUTE for the instance restricted to the hub set $\bar{N}$. This means we select $|\bar{N}|-1$ hubs from the current set $\bar{N}$ and update this set in Line 4. This way, in each iteration we close at least one hub and finally obtain an incremental solution.

Algorithm 5.4: `Decr-MILP` – MILP-based decremental hub selection

Input: A logistics network with hub set N of potential hub nodes, hub bounds L, M
Output: A (L, M)–incremental solution

1 $\bar{N} := N$;
2 **for** $h = M, \dots, L$ **do**
3 compute a solution $(N_h, \mathcal{R}_h)$ to $h-$HUBROUTE for the network restricted to the hub set $\bar{N}$ using Algorithm 5.2;
4 $\bar{N} := N_h$;
5 **return** $(N_L, \mathcal{R}_L), \dots, (N_M, \mathcal{R}_M)$

5.3.2 Speedups for Routing

We will see in the computational study that the decremental framework combined with $p = \Delta(n, \mathcal{R}_h)$ suffers from long running times, the bottleneck being the path-based local search (LS) of the routing algorithm REINIT($\bar{N}$). We present speedup techniques for the subroutine LS. Following the outline of Subsection 2.4.3 it consists of Type A moves that remove a single path at a time, and of Type B moves that remove all paths that use a specific edge. A path phase of Type A moves is succeeded by an edge phase of Type B moves. We propose the following heuristic modifications for speed up:

Edge Heuristic (EH) The Edge Heuristic is used to mark specific edges, that were considered in the edge rerouting phase of the local search before and that did not yield any further improvement. Expecting no further improvement from those edges, we skip marked edges in further iterations of Type B moves.

Commodity Heuristic (CH) Similarly, the Commodity Heuristic marks commodities that did not yield any improvement. We use this heuristic for Type A moves of the local search

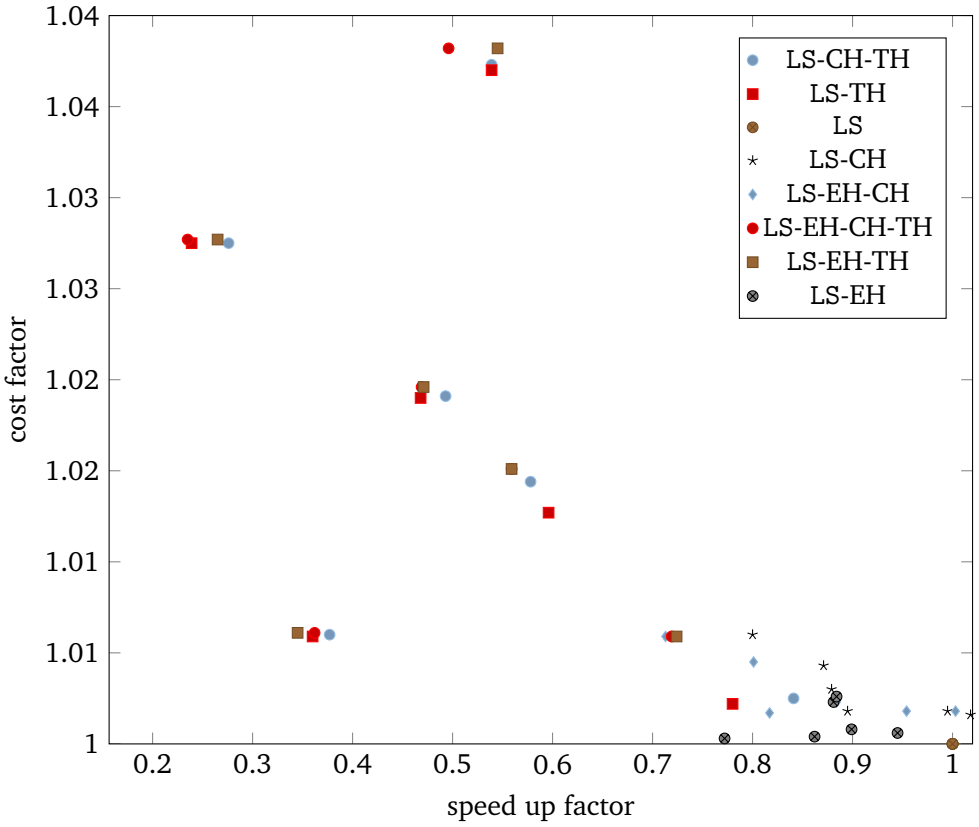


Figure 5.4: Performance of speedup techniques for local search (LS): Combining Edge Heuristic (EH), Commodity Heuristic (CH), and Threshold Heuristic (TH). For detailed numbers see Table A.1 in Appendix A.1.

and skip considering the rerouting of commodities where the previous two rerouting steps did yield any improvements.

Threshold Heuristic (TH) The Threshold Heuristic applies to the edge phase: Before considering an edge, we compute for each commodity using that edge the ratio of partial transportation costs for this commodity and the cost savings if this commodity was removed. If this value lies under a certain threshold, we assume that the commodity already uses this edge effectively. We then decide that this commodity should not be considered for removal, i.e. we do not delete all commodities on this edge but keep those that have a good ratio.

All these measures are heuristic and yield better running times which is often concomitant to a loss in solution quality. Figure 5.4 shows results from preliminary tests on smaller networks. The best solution quality is attained when no speed-up technique is used. Indeed, many combinations yield significant speedups up to a factor of almost three but the solution quality can be up to five percent worse.

As we seek to minimize the impact that suboptimal routing algorithms may have on hub decisions, we chose to disable all speed-up techniques but the Edge Heuristic in further

computations. The latter one is accurate and the speedup factor is just sufficient to solve our larger instances.

Nevertheless, these results encourage further engineering within our algorithmic framework. For example, we suggest to apply all speed-up techniques during initial steps when the impact of routing decisions is minor for hub decisions, and disable one after another, when more accurate routing algorithms are required.

5.3.3 Meta-node Aggregation

Two of our test instances come with a large number of demands, even after applying the aggregation on the basis of common source-sink pairs as described in Subsection 4.7.5. This is a major concern since the running times of our algorithms heavily depend on the number of demands. Remember that a source-sink aggregation limits the number of demands to the number of sources times the number of sinks. So we can reduce the number of demands even further if we can reduce the number of sinks and sources, i.e. if we aggregate sources or sinks into *meta-nodes*. This simple observation leads to the concept of meta-node aggregation for a heuristic contraction of the network.

Here we describe this concept restricted to the set of source nodes but the same aggregation can be applied to the sink side as well.

Let $\mathcal{O}, \mathcal{D}$ be the set of source respectively sink nodes and $J(s), s \in \mathcal{O}$ be the set of demands j that have $\text{source}_j = s$. Let us assume for a moment that we were able to determine groups of demands such that in an optimal solution their source–sink paths connect their sources to a *common* hub node. Then we can jointly decide the source-hub edges of the routing of all demands. Thus the final choice of their connecting hub may be deferred to some hub location algorithm. To make use of this observation, we virtually relocate all demands of a group to a common meta-source during an aggregation phase. Routing these demands on an edge that connects a meta-source with a hub node then corresponds to routing all these demands from their original source to the same hub node in the original instance.

Note that we are actually not able to "guess" the groups that will use the same hub in optimum solutions. So all aggregation algorithms that we present in this section are heuristics and differ in the way they determine these groups. We now give a formal definition of the resulting *meta-instance*

Definition 5.4 (Meta-instance) *Given a set $\hat{\mathcal{O}}$ of meta-sources and an assignment function $\psi : \hat{\mathcal{O}} \to 2^J$ such that $\dot{\bigcup}_{\hat{s}\in\hat{\mathcal{O}}}\psi(\hat{s}) \subseteq J$, we define the resulting* meta-instance $\hat{G} = (\hat{V}, \hat{E})$ *as follows:*

$$\hat{V} := \hat{\mathcal{O}} \cup \left\{ s \in \mathcal{O} : J(s) \setminus \dot{\bigcup}_{\hat{s}\in\hat{\mathcal{O}}}\psi(\hat{s}) \neq \emptyset \right\} \cup \mathcal{D} \cup N$$

and

$$\hat{E} := (\hat{\mathcal{O}} \times N) \cup E \setminus \left\{ (s,n) \in \mathcal{O} \times N : s \notin \hat{V} \right\}.$$

Moreover for the meta-instance we set $\text{source}_j := \hat{s}$ *for all $j \in \psi(\hat{s})$ and all $\hat{s} \in \hat{\mathcal{O}}$.*

We call edges in $\hat{E} \setminus E$ *meta-edges* and use a very simple cost function for these edges. In fact it only takes two values: If the edge is not used at all, it has the value zero. Otherwise it equals

the sum of costs for routing all demands of the incident meta node to the incident hub node.

$$c^T_{(\hat{s},n)}(a) := \begin{cases} 0 & \text{if } a = 0, \\ \sum_{s' \in \mathcal{O}} c^T_{(s',n)} \left(\sum_{j \in J(s') \cap \psi(\hat{s})} d_j \alpha_j \right) & \text{o.w.} \end{cases} \tag{5.53}$$

This reflects our assumption that demands originating at some meta-source $\hat{s}$ in the meta instance are routed jointly on a common meta-edge $(\hat{s}, n) \in \hat{\mathcal{O}} \times N$. This joint routing in the meta-instance corresponds to a routing in the original instance along possibly disjoint edges (source_j, n) for involved demands $j \in \psi(\hat{s})$. Note that we ignore any possible cost sharing with demands not in $\psi(\hat{s})$ with this definition. This implies that any solution to a meta-instance can be transformed into a solution to the original instance of no greater cost but the converse statement is not true: Two demands that are assigned to distinct meta-sources could still belong to the same original source, and if so, routing them on common source hub edge could use economies of scale, that are not reflected in the meta instance.

Finally, we actually reduce the number of demands: We can apply the aggregation on the basis of common source-sink pairs now to the meta-instance. This limits the resulting number of demands to the sum of sources and meta-sources times $|D|$. Thus, when designing algorithms that compute meta-node aggregations, the resulting size of the sets $\hat{\mathcal{O}}$ and the remaining nodes $\bar{\mathcal{O}} := \{s \in \mathcal{O} : J(s) \setminus \dot{\cup}_{\hat{s} \in \hat{\mathcal{O}}} \psi(\hat{s}) \neq \emptyset\}$ mainly influences the final number of aggregated demands. A primary goal is to keep the set $\bar{\mathcal{O}}$ small, but there is also another tradeoff: Any demand that is assigned to a meta-source can no longer be satisfied with a direct connection, possibly a costly restriction. For the set of meta-sources we impose the restriction $|\hat{\mathcal{O}}| \leq |N|$, since we do not see any benefit of having more meta-sources than there are hub nodes.

To the best of our knowledge, the theoretical aspects of graph contractions that significantly reduce the number of nodes are subject of very recent research. We can refer the interested reader to [Däu+17] where a broad theoretical basis for contractions that preserve distances in undirected networks is established and efficient algorithms are presented.

5.3.4 Computing a Meta-node Aggregation

In this section we present two heuristics that compute a set of meta-nodes $\hat{\mathcal{O}}$ together with an assignment function ψ according to Definition 5.4. Both attempt to capture economies of scale on source–hub edges by considering classical location problems. The work presented in this subsection has proof of concept status. The two heuristics enable us to solve our hub location model even for the largest instances in our test set. On the other hand we believe that improvements are still possible, especially for shorter running times, but also to improve solution quality.

The first heuristic, M`-med-agg`, optimizes the assignment of source nodes to hub nodes, with the restriction to use at most M hub nodes and subject to assignment costs governed by transportation costs. The second one, `fac-loc-agg`, considers a classical facility location problem where the set of clients consists of all demands and the facilities correspond to tariff levels on source–hub edges. We now discuss both methods in more detail.

The M`-med-agg` **Heuristic** This aggregation is motivated by networks, where economies of scale on source-hub edges can be hardly realized due to very small demand values. In such

Algorithm 5.5: M-`med-agg` – Meta-node aggregation with k-medians

Input: A logistics network, a bound SourceBound on the number of remaining meta-sources in the meta-instance

Output: A set of meta-sources $\hat{\mathcal{O}}$ together with a function $\psi : \hat{\mathcal{O}} \to 2^J$ that yields a meta-instance according to Definition 5.4

1 Clients $:= \mathcal{O}$, Facilities $:= N$;

2 $c_{ns} := c^T_{(s,n)} \left(\sum_{j \in J(s)} d_j \alpha_j \right) \; \forall n \in$ Facilities, $\forall s \in$ Clients; `// set assignment costs`

3 Solve the k-median instance (Facilities, Clients, c) with $k :=$ SourceBound using standard MILP-software and let KSol := Clients $\to$ Facilities be the resulting assignment;

4 **for** $n \in N$ **do**

5 **if** $\mathsf{KSol}^{-1}(n) \neq \emptyset$ **then**

6 $\psi(n) := \mathsf{KSol}^{-1}(n)$;

7 $\hat{\mathcal{O}} := \hat{\mathcal{O}} \cup \{n\}$;

8 **return** $(\hat{\mathcal{O}}, \psi)$;

a situation it may be beneficial to keep the sum of distances of selected source-hub edges as small as possible.

Note that we have no distance information and distance may only correlate with transportation cost. The idea here is to solve a k-median problem formulated for transportation costs under a *single-sourcing assumption*: For some logistics networks it can be assumed that the restriction to serve all demands of a single source by exactly one hub node does *not* lead to a significant increase in transportation costs compared to allowing full flexibility. Note that this assumption also excludes the possibility to serve any demand via a direct connection. Under a single-sourcing assumption the cost for the transportation of all demands of a source to a hub node can be preprocessed and used as input for our M-`med-agg` heuristic.

The heuristic is outlined as Algorithm 5.5 and allows a parameter SourceBound that controls the number of remaining meta-nodes. We construct a SourceBound-median instance in which all source nodes are clients, all hub nodes are potential facilities and the assignment cost from a client to a facility accounts for the transportation cost from all demands at the source node to the hub node, as defined in Line 2.

After solving this SourceBound-median instance with standard MILP software to optimality we set the assignment function ψ to the inverse of the assignment of source nodes to hub nodes that we obtained as a solution.

The `fac-loc-agg` Heuristic Algorithm 5.5 has two major disadvantages: The underlying single-sourcing assumption may be false and forbidding direct connections can be too restrictive, thus result in highly suboptimal solutions. We believe that the second aspect could be mitigated with a preprocessing of cost efficient direct connections, which can then be removed from the problem instance together with all demands satisfied by them. Up to now, we however lack any convincing a priori arguments to tell cost efficient direct connections from cost inefficient ones. We use a different strategy: We introduce for each node a SourceScore() equal to the cost for sending all its demands to nearby hub nodes. Source nodes with high

Algorithm 5.6: `fac-loc-agg` – Meta-node aggregation with facility location

Input: A logistics network, a bound SourceBound on the number of remaining sources in the meta-instance, and an algorithm F to solve instances of the facility location problem

Output: A set of meta-sources $\hat{\mathcal{O}}$ together with a function $\psi : \hat{\mathcal{O}} \to 2^J$ that yields a meta-instance according to Definition 5.4

1 $\hat{\mathcal{O}} = \emptyset$, $\mathsf{SourceScore}(s) := 0 \; \forall s \in \mathcal{O}$;
2 **for** $s \in \mathcal{O}$ **do**
3 $\quad \mathsf{Clients} := J(s)$, $\mathsf{Facilities} := \bigcup_{e \in \delta^+(s)} K(e)$;
4 $\quad \bar{d}_j(s) := \begin{cases} d_j & \text{if } j \in J(s) \\ 0 & \text{o.w.} \end{cases} \quad \forall j \in J$;
5 $\quad$ **for** $e = (s,n) \in \delta^+(s)$ *with* $n \in N$ **do**
6 $\quad\quad e' := (n, \mathrm{sink}_j)$;
7 $\quad\quad$ **for** $k \in K(e)$ **do**
8 $\quad\quad\quad$ **for** $j \in J(s)$ **do**
9 $\quad\quad\quad\quad c_{kj} := \max_{\pi \in \Pi} d_j \alpha_{j\pi} / \beta_{\pi k} + c^V_{e'}(j, \bar{d}(s))$; `// see Equation (4.49)`
10 $\quad$ use F to solve the facility location instance $(\mathsf{Facilities}, g, \mathsf{Clients}, c)$ and let $\mathsf{FSol} : \mathsf{Clients} \to \mathsf{Facilities}$ be the resulting assignment ;
11 $\quad$ set $\mathsf{SourceScore}(s)$ to the cost of FSol;
12 $\quad \hat{\mathcal{O}}' := \{n \in N : \exists k \in \mathsf{FSol}(\mathsf{Clients}) \text{ with } (s,n) = e(k)\}$;
13 $\quad \hat{\mathcal{O}} := \hat{\mathcal{O}} \cup \hat{\mathcal{O}}'$;
14 $\quad \psi(n) := \psi(n) \cup \{j \in J(s) \text{ with } e(\mathsf{FSol}(j)) = (s,n)\} \; \forall n \in \hat{\mathcal{O}}'$;
15 **for** $(\mathsf{SourceBound} - |\hat{\mathcal{O}}|)$ *many sources* s *with highest* $\mathsf{SourceScore}(s)$ **do**
16 $\quad \psi(n) := \psi(n) \setminus J(s) \; \forall n \in \hat{\mathcal{O}}$; `// undo assignment for` s
17 **return** $(\hat{\mathcal{O}}, \psi)$;

scores are more eligible to be served with direct connections and are therefore kept as original nodes in the aggregation step.

Our second heuristic, `fac-loc-agg`, outlined in Algorithm 5.6, allows for multi-sourcing after the aggregation step. We define an instance of the facility location problem for each source node to compute optimal transportation costs for sending all demands of a source to any nearby hub node. In this instance each demand of the source node is a client and each tariff level on any incident edge is a facility (Line 3). The opening cost of a facility that corresponds to tariff level k is set to its cost g_k, cf. the definition of tariff cost in Equation 5.1.

The cost for assigning client j to facility k, as set in Line 9, is composed of two parts: The first part, $\max_{\pi \in \Pi} d_j \alpha_{j\pi} / \beta_{\pi k}$, accounts for $c^L_e(d_j)$ of the linearized tariff cost from Section 5.2. The second part, $c^V_{e'}(j, \bar{d}(s))$, estimates the cost share of demand j for routing it from hub n along e' to its sink. We use here the weighted volume cost share from Subsection 5.2.7.

Note that without the second part the resulting facility location problems would most likely yield an assignment that sends all demands to the one cheapest hub nearby and thereby result in single-sourcing.

The facility location instances are solved for each source with a generic algorithm F,

either a greedy heuristic or with standard MILP software. The assignment from the solution of each instance is tentatively stored in the return values $(\hat{\mathcal{O}}, \psi)$ and the cost of the solution is assigned as a SourceScore to each source in Lines 11 to 14.

If the input parameter SourceBound allows to keep some of the original sources, then the loop in Line 15 undoes the assignment in ψ for the permitted number of sources, i.e. those that have a high SourceScore.

5.4 Computational Results for Incremental Hub Chains

In this section we evaluate our heuristics to compute incremental hub chains on a set of seven real-world instances, which were provided by 4flow AG [4fl17], a logistics consultancy company serving small, medium-sized, and global customers from a broad spectrum of industries. For all of them, selecting more than 20 hubs is out of scope for practical operations. No opening costs for hub nodes are incurred in these instances.

5.4.1 Test Instances and Testing Environment

The first five instances, H_Auto_1 to H_Auto_5 are the inbound logistics networks of manufacturers in the automotive industry. X_Handel is a network from retail logistics and X_Ersatzteil from spare parts logistics. Table 5.2 lists the sizes that are relevant for the algorithmic complexity of the networks. Here the column "#OD-pairs" denotes the set of source-sink pairs (u, v), such that there is a demand from u to v.

Notice that the number of physical commodities may vary greatly among the instances and many demands with a common source-sink pair may occur for different physical commodities. However, all our heuristics are preceded by a preprocessing that aggregates physical commodities on the basis of common source-sink pairs, see Subsection 4.7.5.

This allows us to identify the OD-pairs with the abstract commodities that are relevant for the performance of our algorithms. All networks share a common structure: The hub network is a complete graph and all sources and sinks are connected with each hub node. A direct connection between a source and a sink is only present if the source-sink pair is also an OD-pair. In all instances the properties mass, volume, and loading meters are relevant.

Table 5.2: Sizes of our test instances

network	sources	#hubs	#sinks	#OD-pairs	edges
H_Auto_1	400	85	23	754	43 849
H_Auto_2	3 082	99	80	12 387	335 127
H_Auto_3	1 212	92	39	2 598	126 062
H_Auto_4	1 412	92	14	1 890	141 454
H_Auto_5	1 287	103	13	3 284	147 690
X_Handel	357	102	13 866	71 230	1 532 278
X_Ersatzteil	2 183	92	5 689	1 321 150	2 053 746

All reported cost values refer to an a posterior evaluation of the solutions by applying c_e^T from Subsection 5.1.1 on the transported commodities on each edge e.

Unfortunately we cannot determine the true price of being incremental, see Definition 5.3, in the networks of our test sets, as we are not able to compute OPT(h) in (5.49). Instead we compare a partial solution $(\bar{N}_h, \mathcal{R}_h)$, $L \le h \le M$ with the best known solution Best(h) and report the maximum as well as the average *observed* price of being incremental. More formally, let Alg.A(h) denote the h-th part of an incremental solution of Algorithm A, then we consider

$$\text{maxGap} := \max_{h=L,\dots,M} \frac{\text{cost}(\text{Alg}.A(h))}{\text{cost}(\text{Best}(h))}, \qquad \text{avgGap} := \frac{1}{M-L+1} \sum_{h=L,\dots,M} \frac{\text{cost}(\text{Alg}.A(h))}{\text{cost}(\text{Best}(h))} \tag{5.54}$$

All algorithms from this chapter have been implemented in C++ and compiled with gcc 4.8.5 on openSUSE 42.1 Linux with kernel 4.1.34-33. Computations have been performed on cluster nodes with two DualCore-Opteron 2218 processors (2.6 GHz, 64 bit) and 16 GB of memory using CPLEX 12.6 for linear and mixed integer programs. A time limit of 9 hours for all MILP computations is imposed, unless reported otherwise, and we allow for a relative optimality gap of 0.2% for shorter running times. Since all computations were conducted on a multi-user environment with shared resources, all provided running times are subject to natural fluctuations. We claim by no means to have succeeded or aimed at establishing a benchmark environment, but our main focus is on low cost solutions in acceptable running times. For the latter concern we list the CPU user time that we have measured. All algorithms are sequential except for calls to CPLEX MILP routines that are allowed to use up to 4 cores in parallel.

5.4.2 Performance of LP-based Hub Search

In a first experiment we consider the two medium size instances with respect to the number of OD-pairs, H_Auto_1 and H_Auto_4. For those instances the LP-based hub search, Algorithm 5.2 from Subsection 5.2.8, can be applied in a reasonable running time for all considered hub bounds and also Algorithm 5.4 has a reasonable running time.

Algorithm 5.2 is based on an LP relaxation and interestingly, it seems to be much more difficult to solve the LP relaxation than enforcing integrality of an LP solution with branch-and-bound: We observed that most of the MILPs from Line 15 are solved at the root node and the gap between the final integral solution and the cost of last LP solution obtained in Line 4 is less than 1‰. We thus conjecture that the LP relaxation to HUBLINPATH tends to be strong for practical instances, which is in contrast to Example 5.1 showing that this gap can be theoretically arbitrary large. On the other hand, the LP-based Hub Search suffers from high degeneracy in the LP formulation, which slows down the primal simplex algorithm especially for large instances in the test set. We therefore cannot report results on the strength of the formulation for larger instances.

As discussed earlier, Algorithm 5.2 is heuristic in many ways. In particular, we have to limit the number of pricing rounds and stop with an incomplete LP relaxation that is unlikely to contain the set of paths needed for an optimum. Thus our empirical observation of very small integrality gaps could be a peculiarity to these incomplete LP relaxations and further research is needed to investigate the actual strength of the LP relaxation HUBLINPATH.

In Figures 5.5 and 5.6 we evaluate Algorithm 5.2 on the test instances H_Auto_1 and H_Auto_4. The exact cost values can be found in Tables A.2 and A.3 respectively in the

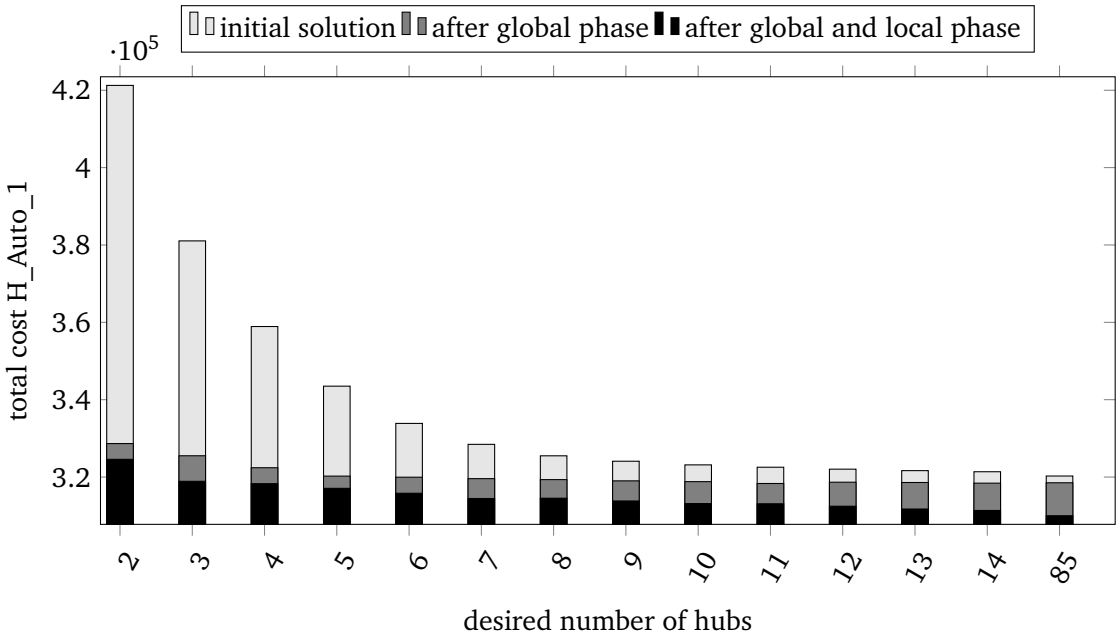


Figure 5.5: Performance of global and local phase of Algorithm 5.2 on H_Auto_1, for detailed numbers see Table A.2.

appendix. We run Algorithm 5.2 for each hub bound in $\{2,\ldots,14,|N|\}$ and compare the optimum cost for a) the initial LP after Line 1 b) for the LP after the global phase from Lines 5–7 and c) for the LP after the local phase in Lines 8–14. We do not report running times here but mention, that Algorithm 5.2 terminated within at most 4 hours for all instances.

We can observe that smaller hub bounds lead to increased costs of the initial LP solution. This is however compensated by the global phase: The costs of the solution after the global phase grows only slowly with decreasing hub bounds. The cost reduction of the local phase shows no specific dependence on the hub bounds, but especially for larger hub bounds, the local phase accounts for more cost savings than the global phase.

Though not tested rigorously, we also considered a variant where the order of both phases is inverted: The result is that the local phase now accounts for most of the cost improvement whereas applying the global phase second has almost no effect. Even though resulting solutions do not differ much for large hub numbers, for smaller hub numbers, the resulting costs are considerably higher. Thus we do not report this variant here. An explanation could be that the local phase leads hub decisions towards a local optimum from which the global phase cannot recover afterwards.

5.4.3 Incremental Solutions

In the following experiments try to observe a price of being incremental for H_Auto_1 and H_Auto_4: For both instances we know good non-incremental solutions obtained for each hub bound by Algorithm 5.2, abbreviated `hub-mip-heur` in this subsection.

We now summarize preliminary tests for incremental solutions: We have first tested decremental algorithms of the framework `Decr-Greedy`(p,A) choosing for $p(n,\mathcal{R})$ the functions

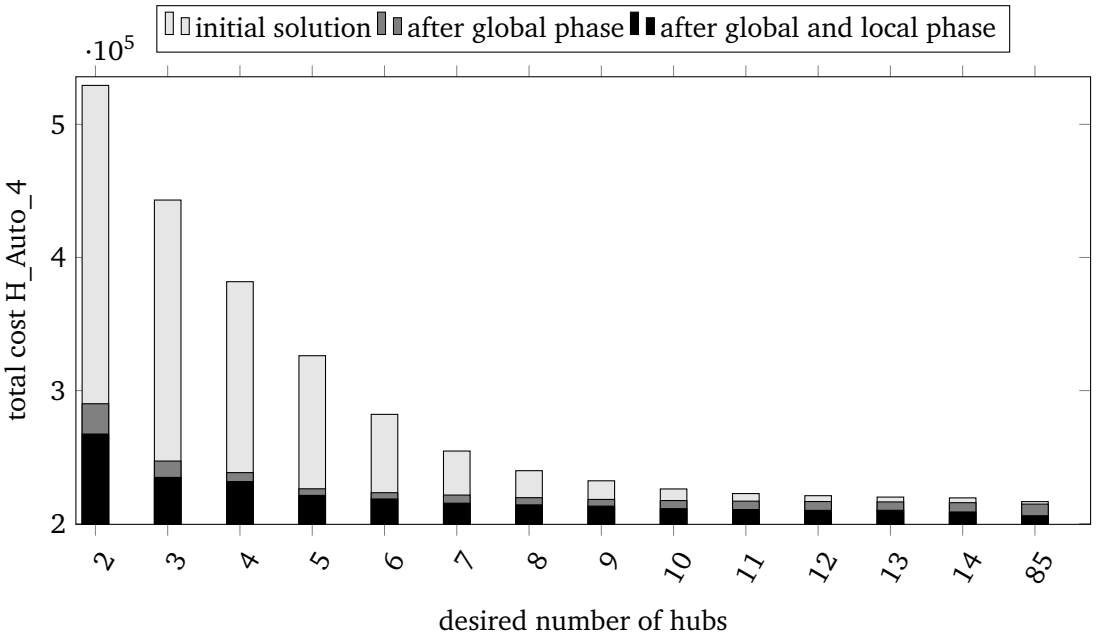


Figure 5.6: Performance of global and local phase of Algorithm 5.2 on H_Auto_1, for detailed numbers see Table A.3.

$\text{USAGE}_\pi(n, \mathcal{R})$ from (5.50) for relevant properties π, and also $\text{USAGE}_\#(n, \mathcal{R})$ from (5.51). Combined with the choices $A =$ `hub-mip-heur` and $A =$ `REINIT`$(\bar{N})$ from Subsection 5.2.6 this yields eight configurations.

We observe that the impact of using different variants for USAGE is minor: All variants have similar running times and and solution cost. For each property π there is a hub number, for which the variant using USAGE_π leads to the best solution when compared to using the other properties.

On average the properties loading meter (ldm) and weight (wgt) slightly outperform the property volume and also the variant $p(n, \mathcal{R}) = \text{USAGE}_\#$. Thus when searching for a price of being incremental we restrict to report only configurations using these two properties. We also include Algorithm 5.4, `Decr-MILP`, as the only decremental algorithm that uses a MILP-subroutine to decide which hub to close next. All reported configurations are subsumed in Table 5.3. The results are visualized in Figures 5.7 and 5.8.

Incremental Solutions for H_Auto_1 and H_Auto_4

When testing the decremental algorithms on H_Auto_1 we find domination effects, especially for the configurations `greedy-ldm-rmls` and `greedy-weight-rmls`: Sometimes we encounter a solution for a hub bound k that has less cost than a solution for a bound $k+1$. When this happens, we replace the solution for hub bound $k+1$ with the one found for hub bound k as this still leads to an incremental hub chain. This explains that some of the cost values in Figure 5.7 stay on a pleateau when considering increasing hub bounds. For example the solution for 11 hubs for `greedy-weight-rmls` (algorithm 4) dominates all solutions for larger hub bounds of that algorithm.

Table 5.3: Solver configurations tested for incremental solutions

name	**configuration**
`greedy-ldm-ssp`	Algorithm 5.3, `Decr-Greedy`(p,A) using $A =$ `REINIT` and $p =$ $\text{USAGE}_{\text{ldm}}$
`greedy-ldm-rmls`	Algorithm 5.3, `Decr-Greedy`(p,A) using $A =$ `hub-mip-heur` and $p = \text{USAGE}_{\text{ldm}}$
`greedy-weight-ssp`	Algorithm 5.3, `Decr-Greedy`(p,A) using $A =$ `REINIT` and $p =$ $\text{USAGE}_{\text{wgt}}$
`greedy-weight-rmls`	Algorithm 5.3, `Decr-Greedy`(p,A) using $A =$ `hub-mip-heur` and $p = \text{USAGE}_{\text{wgt}}$
`hub-mip-heur`	use Algorithm 5.2 for each hub bound independently
`mip-close`	Algorithm 5.4: `Decr-MILP`

Since both configurations, `greedy-ldm-rmls` and `greedy-weight-rmls`, differ from `greedy-ldm-ssp` and `greedy-weight-ssp` only in the choice of the routing algorithm A, we conclude that $A =$ `hub-mip-heur` is a strictly weaker routing algorithm than $A =$ `REINIT`$(\bar{N})$ and this is manifested in the experiments on both instances when comparing the respective variants for larger hub bounds. We mention here once more that `hub-mip-heur` relies on several relaxations when modeling transportation costs whereas `REINIT`$(\bar{N})$ works with the exact cost functions.

Interestingly, the configurations `greedy-ldm-ssp` and `greedy-weight-ssp` lead to hub chains that contain costly configurations for small hub bounds: we can observe this in Figure 5.7 for four and six hubs, and more clearly in Figure 5.8 for 11 and 12 hubs. Configurations `greedy-ldm-rmls` and `greedy-weight-rmls` also show this effect but at smaller hub bounds. It seems that the inaccurate routing with $A =$ `hub-mip-heur` delays bad hub closing decisions of the `Decr-Greedy`(p,A) framework. This deficiency of algorithms that are based on USAGE-type functions was the main motivation for a refinement towards $p(n,\mathcal{R}) = \Delta(n,\mathcal{R}_h)$ from (5.52) that we test in Subsection 5.4.4.

The MILP-based heuristics `hub-mip-heur` and `mip-close` yield constantly similar costs on both test instances with two exceptions, hub bounds two and four on H_Auto_4: Here the heuristic `hub-mip-heur` has a gap 8.09 % to the cost of `mip-close` for two hubs, which is surprising since `hub-mip-heur` is not restricted to produce an incremental solution. Both algorithms solve an instance of $2-$HUBROUTE by solving a formulation of HUBLINPATH for this task, but `hub-mip-heur` uses the version of HUBLINPATH that contains all potential hubs, whereas `mip-close` in Line 4 of Algorithm 5.4 uses a formulation of HUBLINPATH that is restricted to the three hubs that were present in the previous iteration. This of course boosts the performance of Algorithm 5.2, which could explain why `mip-close` outperforms `hub-mip-heur` in this situation.

Inspecting Figure 5.8 more closely, we find that `hub-mip-heur` does indeed beat `mip-close` for hub numbers five to fourteen. Doing calculations in Table A.5 we can bound this difference to at most 2%, which is almost attained for hub bound 14. Since `mip-close` computes incremental solutions, this bound also holds for the price of being incremental that we could observe in our experiments. For both H_Auto_1 and H_Auto_4 the heuristics with

MILP-based hub decisions produce solutions with average gaps of less than 1%, whereas the combinatorial heuristics `greedy-ldm-ssp` and `greedy-weight-ssp` have average gaps of 7% on H_Auto_4. However, they offer a very short running time, up to 10 times faster than the `mip-close`. For `hub-mip-heur` the running time is not reported, but can be bounded by 4 hours whereas the average is roughly two hours.

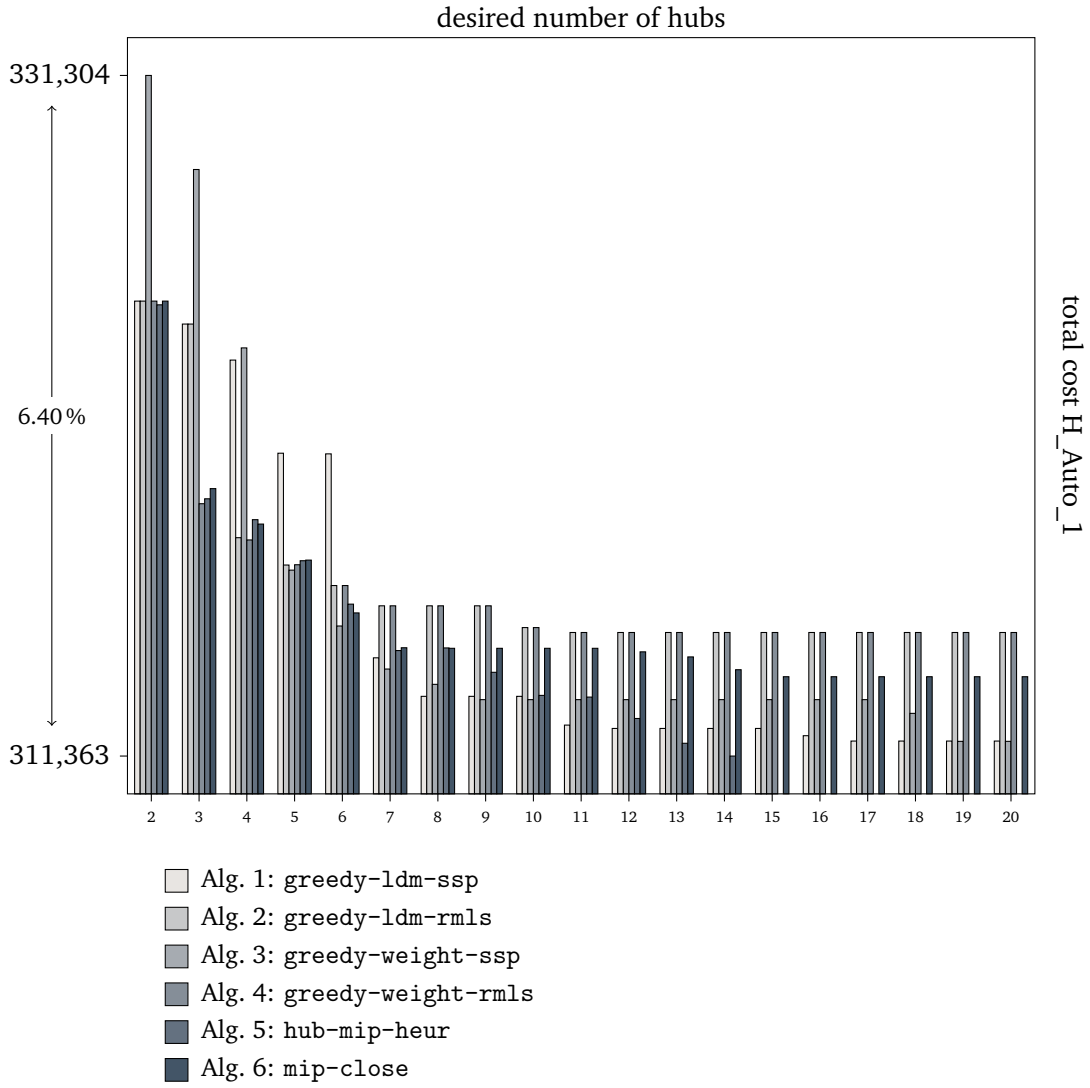


Figure 5.7: Comparisson of incremental and non-incremental solutions on H_Auto_1, for detailed numbers see Table A.4

Table 5.4: Running time and average solution quality of incremental and non-incremental solutions for H_Auto_1.

		Alg. 1	Alg. 2	Alg. 3	Alg. 4	Alg. 5	Alg. 6
running time	[min]	7.6	133.7	7.7	132.1	—	76.5
maxGap	[%]	1.66	1.65	3.07	1.16	0.45	0.81
avgGap	[%]	0.35	0.79	0.51	0.70	0.14	0.46

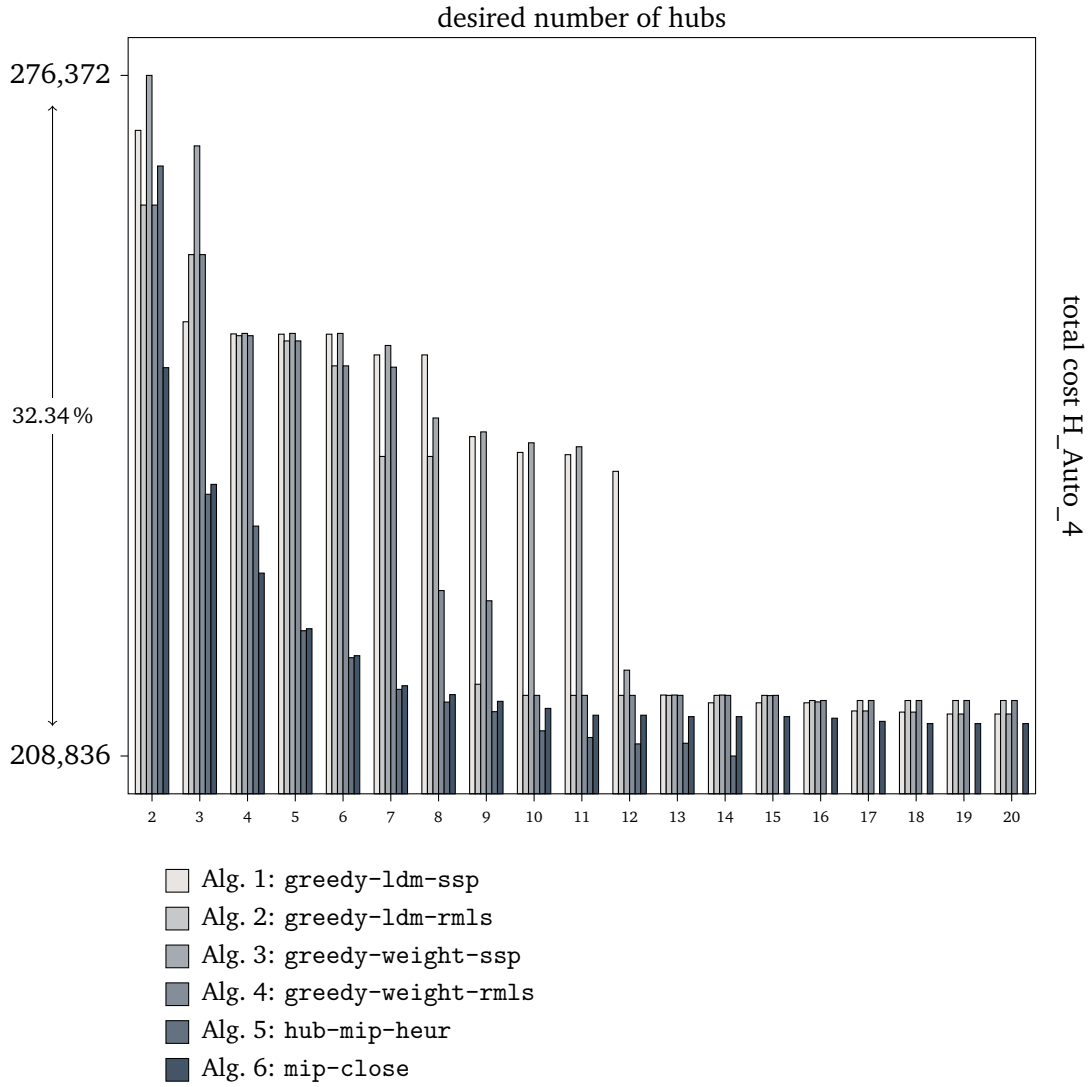


Figure 5.8: Comparisson of incremental and non-incremental solutions on H_Auto_4, for detailed numbers see Table A.5

Table 5.5: Running time and average solution quality of incremental and non-incremental solutions on H_Auto_4.

		Alg. 1	Alg. 2	Alg. 3	Alg. 4	Alg. 5	Alg. 6
running time	[min]	24.5	1542.3	21.4	1497.1	—	262.6
maxGap	[%]	16.08	13.25	15.84	14.84	8.09	1.87
avgGap	[%]	7.73	4.94	7.71	5.03	0.78	0.43

5.4.4 Fast and Improved Combinatorial Heuristics

Concluding Subsection 5.4.2 and 5.4.3, we see that the combinatorial heuristics have the advantage of using the exact functions for tariff cost but the the score functions p used for closing hubs may lead to bad choices for small hub bounds. These result in sudden cost jumps that occur at certain hub bounds, which can however be avoided by MILP-based heuristics. Chances are that the improved combinatorial heuristic `cost-close` from (5.52) that uses $A = \texttt{REINIT}(\bar{N})$ and $p(n, \mathcal{R}) = \Delta(n, \mathcal{R}_h)$, can also avoid these cost jumps.

In the following experiment we extend the test set to instances H_Auto_1 – H_Auto_5. The heuristic `hub-mip-heur` can not handle instances bigger than H_Auto_1 and H_Auto_4, so on other instances we can no longer observe a price of being incremental. Instead we focus on a comparison of `greedy-ldm-ssp`, `cost-close`, and `mip-close`.

H_Auto_1 On this instance `cost-close` beats `mip-close` for almost all hub bounds and even `hub-mip-heur` on several hub bounds, as we can observe in Figure 5.9. The fact that `cost-close` uses the exact cost function, wheras `mip-close` and `hub-mip-heur` use only approximative functions, could be the dicisive advantage. Also Table 5.6 reveals that the running time of `cost-close` is less than two times slower than the one of `greedy-ldm-ssp`.

H_Auto_2 Among the test set of this subsection, H_Auto_2 is the instance with the highest number of OD-pairs. On this test instance the `mip-close` heuristic failed: It hit the time limit before solving the root relaxation of the first hub selection round. We only compare `greedy-ldm-ssp` versus `cost-close` in Figure 5.10. The faster heuristic `greedy-ldm-ssp` produces always a higher cost than `cost-close` with the exception of very few hub bounds, but the difference in cost for these exceptions is always below 0.4%. Sudden cost jumps for `greedy-ldm-ssp` that `cost-close` avoids occur for example between hub bound five and four and between bound 13 and 12. For this instance the height of those jumps is below 1.3%, but then `cost-close` also has one cost jump between hub bound eight and seven. Unfortunately, we cannot decide whether this jump is due to heuristic deficiency or a price to pay for better solutions at larger hub bounds.

Interestingly the running times do not differ much, although the hub closing decision is computationally much more involved for `cost-close`. Both heuristics have a running time of more than twelve hours which is on the borderline between acceptable and unacceptable.

H_Auto_3 The instances H_Auto_3 together with H_Auto_5 are the largest instances that could be solved with the variant `mip-close`. Unfortunately, the decremental steps for large hub numbers are responsible for most of the running time of this heuristic. Thus we restricted to the hub bounds $\{3, \ldots, 13, 16, 20\}$ and report this variant for `mip-close`. In preliminary tests we could observe that this does not harm the solution quality for the considered hub bounds but yields a significant speed-up outperforming the `greedy-ldm-ssp` on this instance and also on H_Auto_5.

We mention that this idea can also be applied to `greedy-ldm-ssp` to a limited extent: Closing for example two hubs at each iteration in Line 4 of Algorithm 5.3 does not lead to much worse solutions. However, it increases the height of cost jumps slightly. Whether applying it to `cost-close` results in worse solutions remains for future research. Since

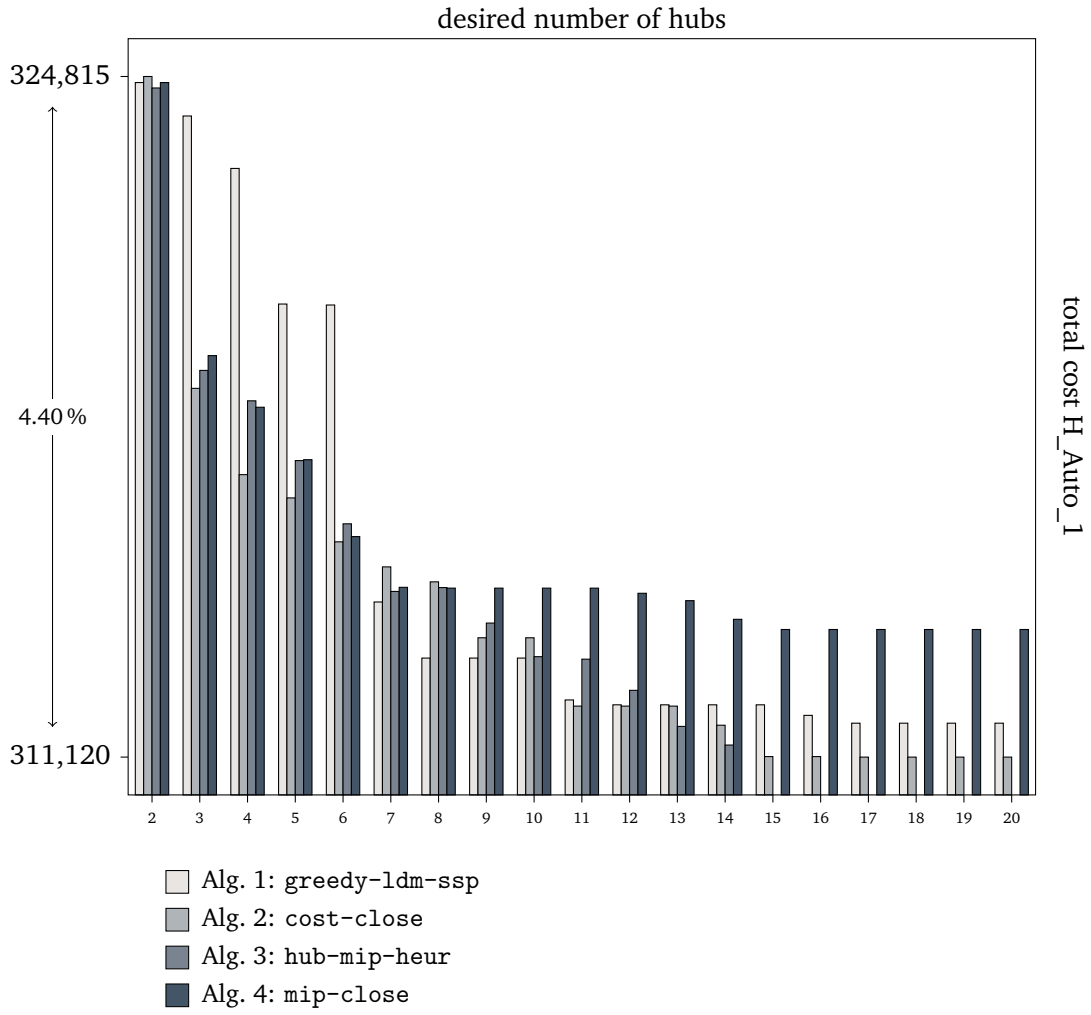


Figure 5.9: Evaluation of `cost-close` on H_Auto_1, for detailed numbers see Table A.4.

Table 5.6: Comparison of running time and average solution quality of `cost-close` on H_Auto_1

		Alg. 1	Alg. 2	Alg. 3	Alg. 4
running time	[min]	7.2	11.4	—	76.5
maxGap	[%]	1.94	0.49	0.47	0.83
avgGap	[%]	0.44	0.07	0.16	0.55

the running time of `cost-close` exceed 36 hours, a speed-up becomes a necessity for this heuristic to be practical.

When comparing solution quality of all three heuristics in Figure 5.11 and Table 5.8, we find that the solution of `greedy-ldm-ssp` beats `cost-close` for hub numbers larger

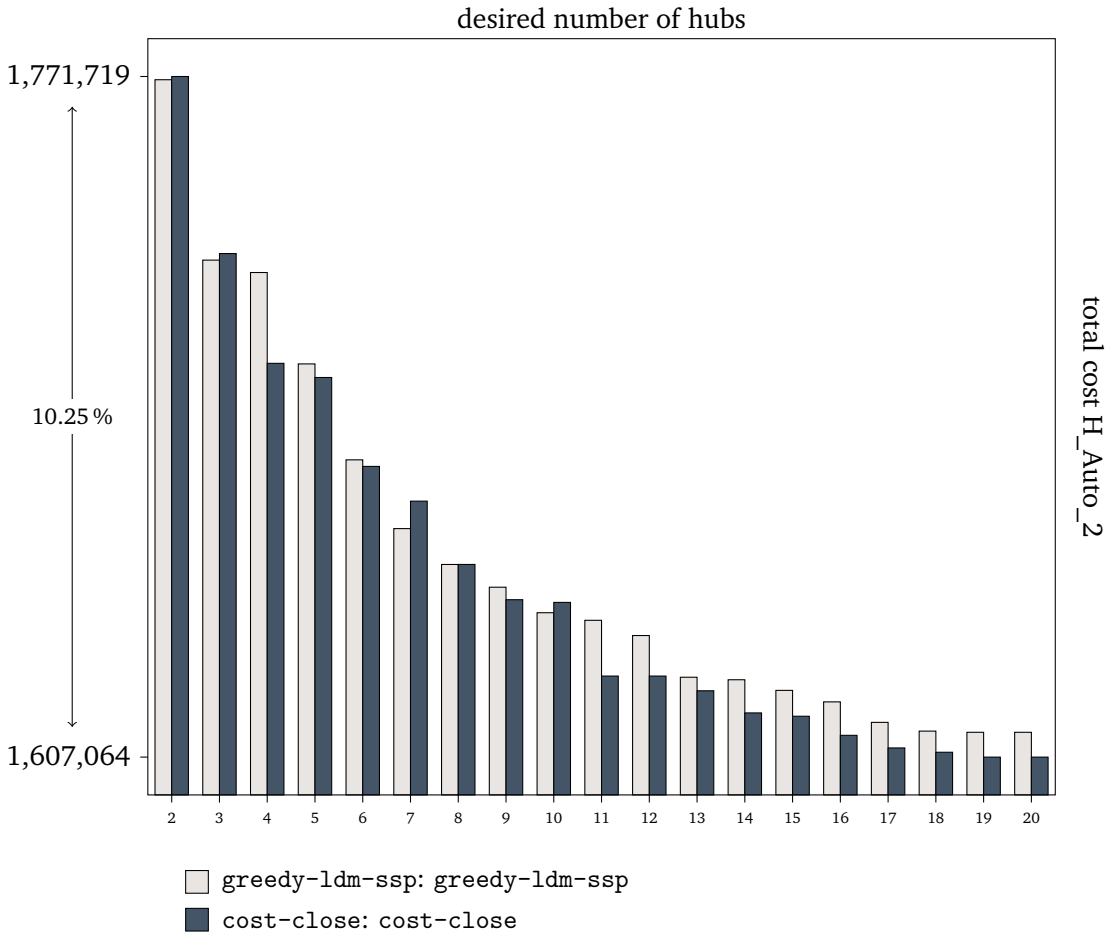


Figure 5.10: Evaluation of `cost-close` on H_Auto_2, for detailed numbers see Table A.7.

Table 5.7: Comparison of running time and average solution quality of `cost-close` on H_Auto_2

		`greedy-ldm-ssp`	`cost-close`
running time	[min]	741.7	880.8
maxGap	[%]	1.29	0.40
avgGap	[%]	0.33	0.04

than ten, whereas for the small bounds two to four, `cost-close` is better. The heuristic `greedy-ldm-ssp` has a smaller average gap and H_Auto_3 is the only instance where we can observe this situation. The solution of the `mip-close` heuristic dominates both of them for the computed hub bounds, except for five to eight hubs, where it has gaps of up to 0.5%.

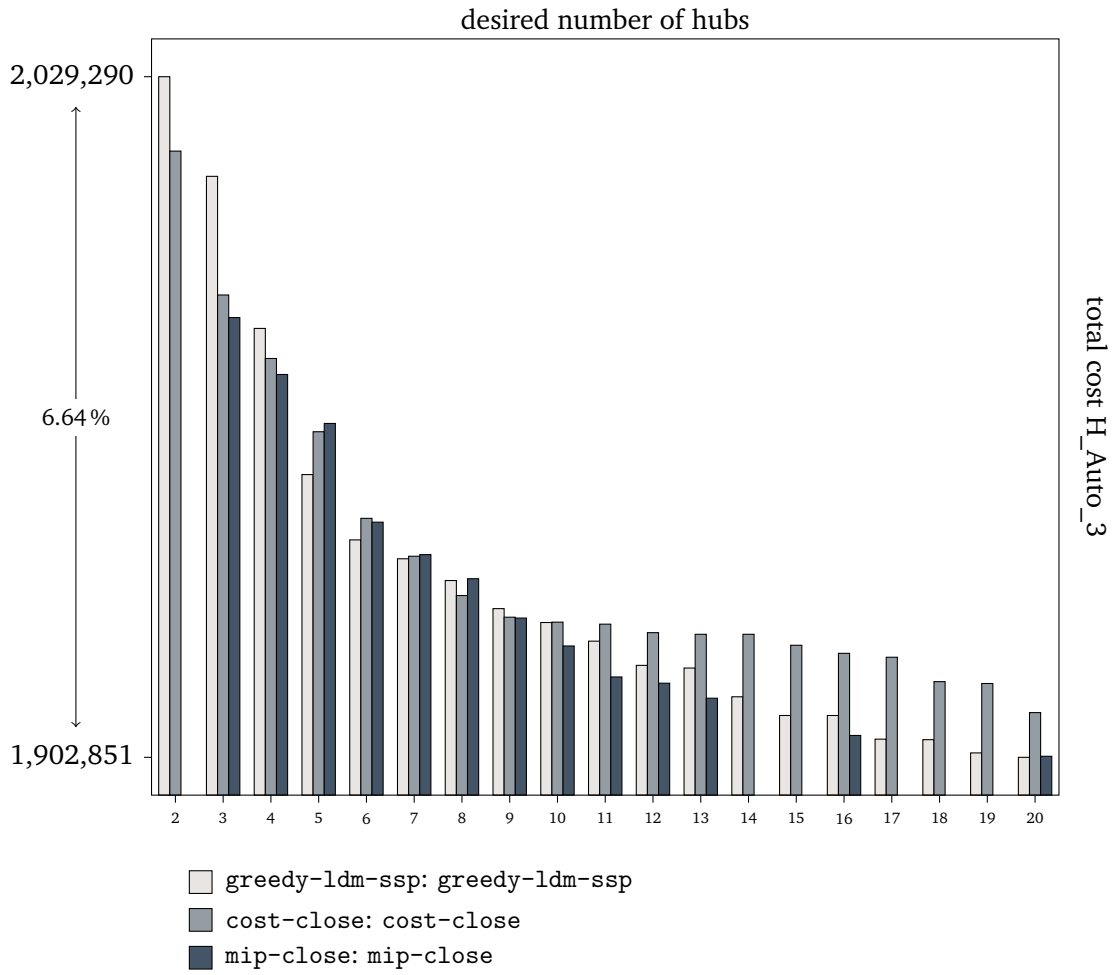


Figure 5.11: Evaluation of `cost-close` on H_Auto_3

Table 5.8: Comparison of running time and average solution quality of `cost-close` on H_Auto_3, for detailed numbers see Table A.7

		`greedy-ldm-ssp`	`cost-close`	`mip-close`
running time	[min]	274.6	2192.4	395.7
maxGap	[%]	1.32	0.80	0.49
avgGap	[%]	0.21	0.39	0.07

H_Auto_4 On H_Auto_4 the best solutions for single hub bounds are found mostly by `hub-mip-heur`, see Figure 5.12. Exceptions are hub bound two, where this heuristic fails, and hub bounds larger than 14, where we did not apply `hub-mip-heur` because of time limitations. The heuristics `mip-close` and `cost-close` compute incremental solutions that let us observe a price of being incremental of roughly 2 %. Here `mip-close` computes a better incremental solution but `cost-close` is less than 1% off on average. Also `cost-close` is roughly only 25% slower than the fastest heuristic `greedy-ldm-ssp` whereas `mip-close` needs ten times longer to terminate, compare with Table A.9. On this instance `cost-close` represents a reasonable alternative to the `mip-close` heuristic.

H_Auto_5 As mentioned before, we only report the faster variant of `mip-close` that is restricted to the hub bounds $\{3, \ldots, 13, 16, 20\}$. In Figure 5.13, we see that each of the three heuristic solutions has some hub bound, for which it has the best cost value and `cost-close` represents a good compromise for solution quality: It attains the lowest relative gap on average. Unfortunately its running time is almost three days and therefore impractical.

Concluding Fast and Improved Combinatorial Heuristics

On the test instances H_Auto_1 to H_Auto_5 the combinatorial heuristic `cost-close` can avoid most of the cost jumps, that occur for the `greedy-ldm-rmls` heuristic but the computational overhead of the score function $p(n, \mathcal{R}) = \Delta(n, \mathcal{R}_h)$ is significant, especially for larger instances, which renders this heuristic impractical. The solutions of `cost-close` are also close to `mip-close` solutions in terms of cost. For very large instances like X_Handel or X_Ersatzteil both methods are not suited: `mip-close` suffers from degeneracy and memory issues and `cost-close` is too slow due to the large number of OD-pairs in these instances. In the following subsections we thus test our aggregation methods aiming at solving larger instances.

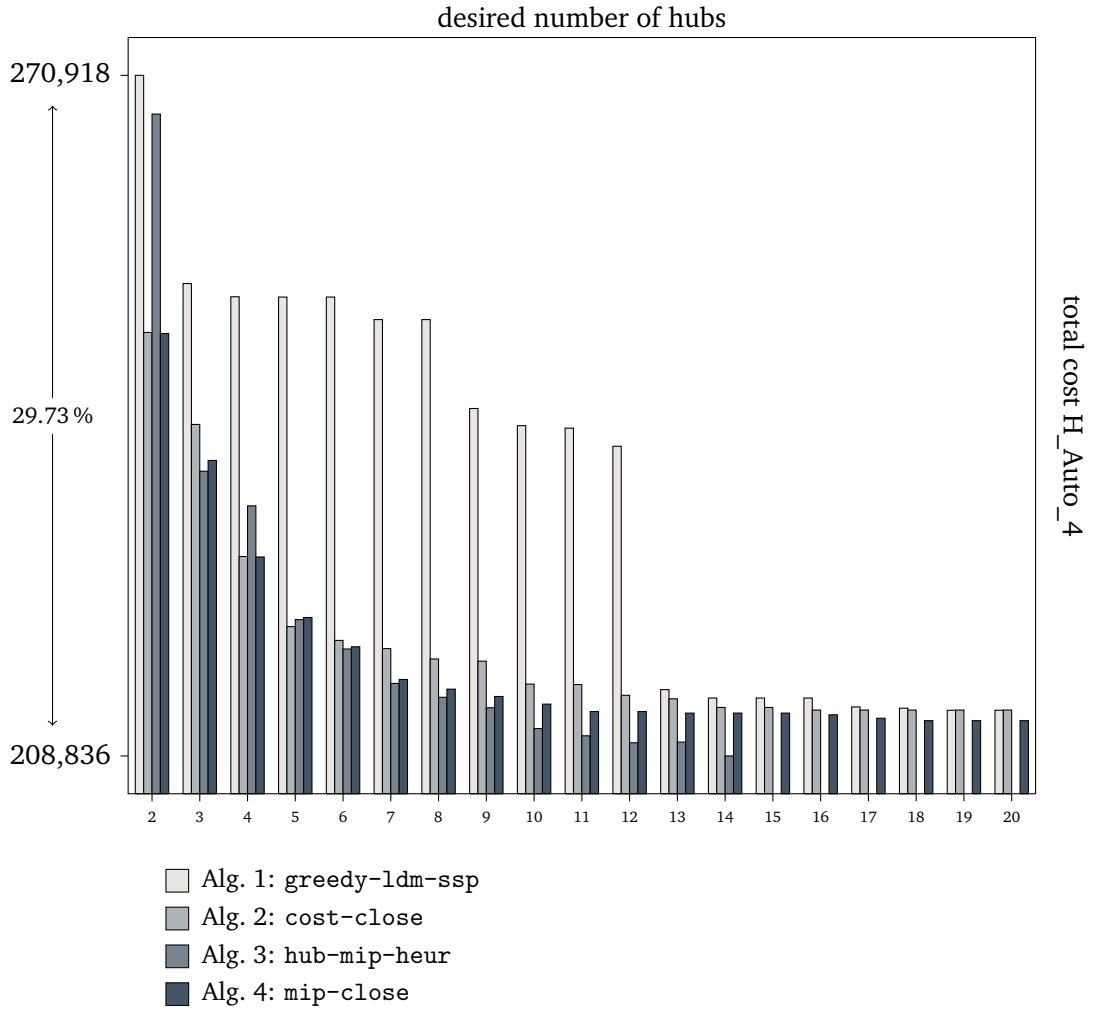


Figure 5.12: Evaluation of `cost-close` on H_Auto_4, for detailed numbers see Table A.9.

Table 5.9: Comparison of running time and average solution quality of `cost-close` on H_Auto_4

		Alg. 1	Alg. 2	Alg. 3	Alg. 4
running time	[min]	22.1	26.8	—	262.6
maxGap	[%]	16.08	2.22	8.09	1.87
avgGap	[%]	7.75	1.04	0.80	0.45

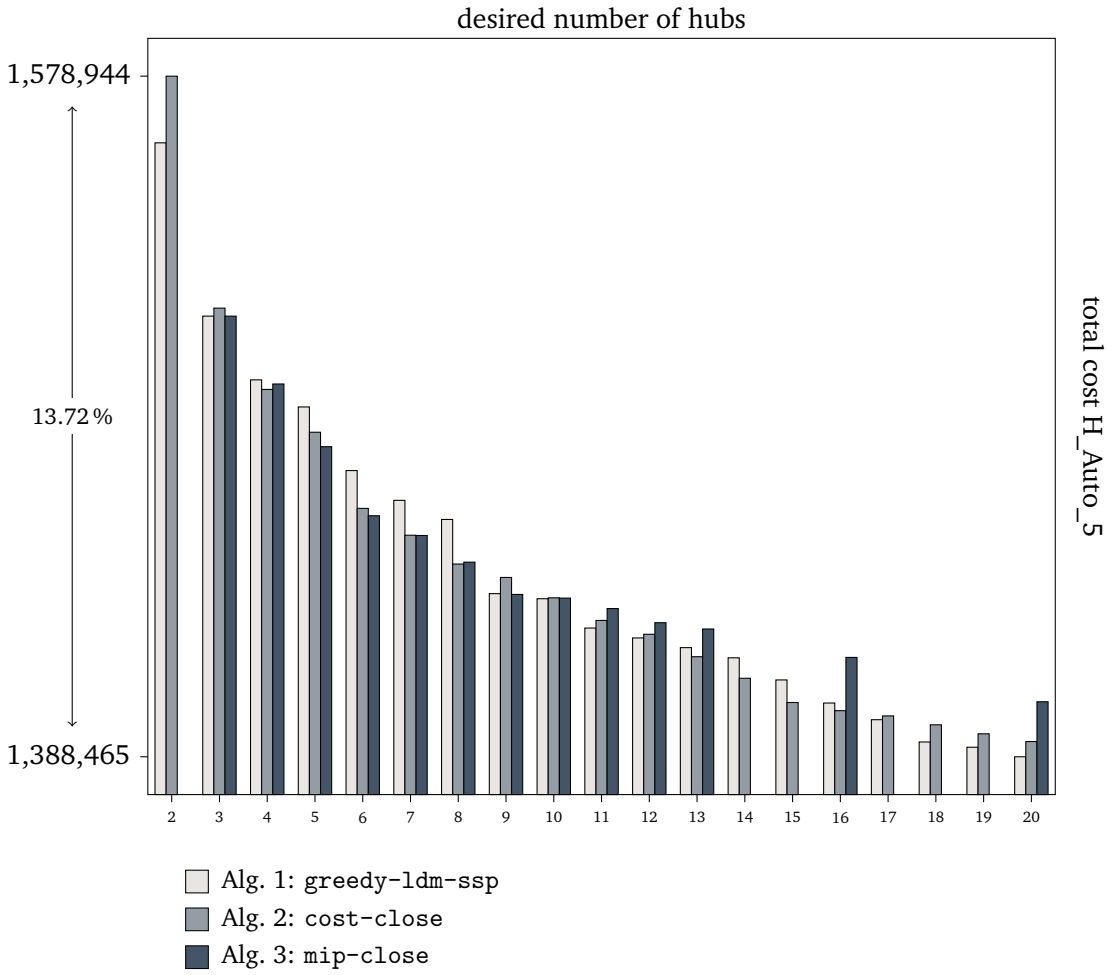


Figure 5.13: Evaluation of `cost-close` on H_Auto_5, for detailed numbers see Table A.10.

Table 5.10: Comparison of running time and average solution quality of `cost-close` on H_Auto_5

		Alg. 1	Alg. 2	Alg. 3
running time	[min]	387.1	4309.1	632.2
maxGap	[%]	0.87	1.20	1.11
avgGap	[%]	0.24	0.18	0.27

5.4.5 Aggregation with `fac-loc-agg` for H_Auto_1 to H_Auto_5

In this subsection, we test Algorithm 5.6, the `fac-loc-agg` heuristic from Section 5.3.4, on the set of instances H_Auto_1 – H_Auto_5.

Since a meta-node aggregation of an instance imposes further restrictions to the original problem, the solution quality can only decrease even if some optimal algorithm was applied to the aggregated instance. The hope is a significantly shorter running time. Our primary interest here is understanding this trade-off. We already know that the `cost-close` heuristic achieves acceptable solution quality on all instances in the test set but can have long running times: It is thus interesting whether this drawback can be remedied by using our aggregation techniques. For this reason we conduct all tests for aggregation techniques only with the `cost-close` heuristic.

All instances from this test set are inbound logistics networks, that is, the number of sinks is much smaller than the number of sources. For this reason the aggregation technique is applied for the sources side only.

For the parameter SourceBound, which controls the desired number of remaining sources, we test all values in $\{200, 300, 400\}$. The second parameter of `fac-loc-agg` is an algorithm for facility location and we test two versions: First, solving a canonical MILP-formulation with standard MILP software, denoted `facLocMilp` and second, using a primal-dual heuristic algorithm from [WS11, Section 7.6, pp. 183], denoted `facLocGreedy`. Together we obtain six combinations of the `fac-loc-agg` heuristic, denoted by `fac-loc-agg-`⟨`facLocGreedy` or `facLocMilp`⟩-SourceBound.

We compare the solutions for aggregated instances with the results of `cost-close` applied to the original instance. The results are very similar throughout the test set and we only summarize the main insights. Detailed figures and tables can be found in Appendix A.1.4.

The first insight is that the smaller the number of remaining sinks in the aggregated instance, the larger is the gap towards the solutions on non-aggregated instances. The best aggregations are always obtained by the configuration `fac-loc-agg-facLocMilp-400`, except for instance H_Auto_1: This instance has only 400 sources and hence `fac-loc-agg-facLocMilp-400` was not tested. Depending on the instance this gap varies between less than 1% on average and 3.29% on average. The worst gaps are attained on H_Auto_2, which has also the largest number of sources.

The second insight is that the cost of `fac-loc-agg` solutions rise only slowly when the hub number decreases. There is an instance dependent hub number such that for hub bounds below this number, the costs rise considerably faster.

This lets us hope that the errors introduced by the restrictions of our aggregation remain rather constant and do not lead to very bad hub decisions for some of the hub bounds. This should be clarified by further research, i.e. whether the gaps of a fixed hub selection could be reduced by disaggregating the instances and restarting a local search for this set of hubs.

Third, the aggregation techniques achieve significant speedups, especially the variant using `facLocMilp` applied to the larger instances H_Auto_3 and H_Auto_5. Here even the variant `fac-loc-agg-facLocMilp-400` has speedup factors of 30 and 40 respectively. The variant `facLocGreedy`, that we chose for its simplicity, shows very weak results: Its solution quality ist is mostly 2–4 % worse on average and for the maximum gap.

It is also surprisingly slower than `facLocMilp` for all but very few configurations which is however due to a detail of the implementation: For speedup reasons we introduced for

facLocMilp a tight timing threshold for solving the facility location instances and some of the larger instances terminate with considerable integrality gaps. The running time of facLocGreedy however can not profit from timing restriction on possibly large facility location instances.

Aggregation with fac-loc-agg on Large Instances

The largest instances in the test set are X_Handel and X_Ersatzteil, which have both a large number of sources and sinks. Thus the fac-loc-agg heuristic is applied to both sides, first to the sources and then independently to the sinks.

For X_Handel we report all combinations of source and sink bounds in $\{200, 300\}$. These are combined with both variants facLocGreedy and facLocMilp as in the preceding subsection. We have also conducted the test for the bound 400 but in contrast to the observations before they consistently yield worse results; so we do not report them.

On X_Ersatzteil the facLocMilp heuristic was not able to solve some of MILP formulations for facility location problems due to memory restrictions. This can possibly be fixed by keeping the affected source or sink node as an original node in the aggregated instance, but this requires further investigation. Here we only test facLocGreedy on X_Ersatzteil with a hub bound of 400 but the resulting set of commodities is still too large for the cost-close heuristic to terminate within a time limit of 8 days for most variants. Only two variants terminated within this time and their results are reported. All computational results can be found in Appendix A.1.4.

On X_Handel we see that facLocGreedy is now faster than facLocMilp when comparing the same node bounds, but again facLocMilp has better gaps on average and for the maximum. Comparing gaps of facLocGreedy and facLocMilp for the same node bounds, their difference does not exceed 4%. The fac-loc-agg-MILP-300-200 has the best gap on average. On X_Ersatzteil some variants are better for larger hub bounds and others for smaller hub bounds. Interestingly, the variant fac-loc-agg-greedy-300-200 is the best on average and even slightly better than fac-loc-agg-greedy-400-200 that leaves more source nodes. Additionally it requires only half of the computation time.

5.4.6 M-med-agg Aggregation

In this subsection we test the M-med-agg aggregation Algorithm 5.5 on X_Handel and X_Ersatzteil. In preliminary tests we considered the instances H_Auto_1 – H_Auto_5 as well but resulting solution costs were roughly 1.5 times higher then the best known. We suspect that the single sourcing assumption is does not apply to these instance and we do not report exact results here.

The outline of the test is identical to the previous tests from Subsection 5.4.5 but we replace facLocMilp and facLocGreedy with M-med-agg. For M-med-agg the role of the input parameter SourceBound is slightly different than for fac-loc-agg: If we allowed for more sources in the aggregated instance than there are hub nodes, then every original source would effectively be assigned to a closest hub. This is not the intention of applying M-med-agg and thus we rather test node bounds $\{40, 70, 102\}$ for X_Handel and $\{12, 40, 92\}$ for X_Ersatzteil. We summarize our results here; further details can be found in Appendix A.1.5.

On X_Handel the solution quality varies only slightly: More than half of the configurations have an average gap of less than 1%. Unfortunately we can see no hints of what makes a good and a bad configuration. The running times differ by factors between two and four and correlate positively with the sink bound. This is no surprise as X_Handel has almost 40 times more sinks than sources.

Also on X_Ersatzteil all variants are very similar concerning solution quality and *M*-med-agg-12-40, *M*-med-agg-40-92 and *M*-med-agg-92-40 achieve the best results. We also mention that the configuration *M*-med-agg-12-12 does *not* yield the best results for the hub bound 12 but it has the best cost for hub bounds six and seven. The running time correlates roughly with the product of source and sink bound.

5.4.7 Comparison on Large Instances

In each of the Subsections 5.4.5 and 5.4.6 we tested one aggregation technique for large instances. In this section we compare both techniques. All heuristics here refer to one configuration described earlier and each one has been selected because it represents one of the best results from a previous test.

X_Handel For X_Handel we can still apply the heuristics `greedy-ldm-ssp` and `greedy-weight-ssp` on the original instance. As depicted in Figure 5.14 and Table 5.11, the greedy heuristics contribute very good solutions and they complement each other: The solutions of `greedy-ldm-ssp` are better for larger hub bounds, and `greedy-weight-ssp` for smaller hub bounds. On average they are slightly outperformed by the `fac-loc-agg`-MILP-300-200 configuration that also achieves a speedup factor of four. The *M*-med-agg-102-40 and *M*-med-agg-40-70 configurations are considerably faster with speed-up factors of 40. Unfortunately these configurations loose roughly one percent of average solution quality.

X_Ersatzteil On X_Ersatzteil most of the previous configurations either hit a time limit of 8 days or suffer from other computational difficulties related to mixed integer programming like memory issues. Thus we compare the two best variants of *M*-med-agg versus `facLocGreedy` in Figure 5.15 and Table 5.12. We find that *M*-med-agg heuristic clearly dominates the `facLocGreedy` heuristic that has a 12% gap on average and is up to 70 times slower than *M*-med-agg. The poor performance of Algorithm 5.6 could be due to the fact that both drawbacks of this heuristic are met by X_Ersatzteil: The large instance size prevents applying the better subroutine `facLocMilp` and imposes small bounds for the remaining number of sinks and sources. On the other hand the single sourcing assumption seems to apply to this network.

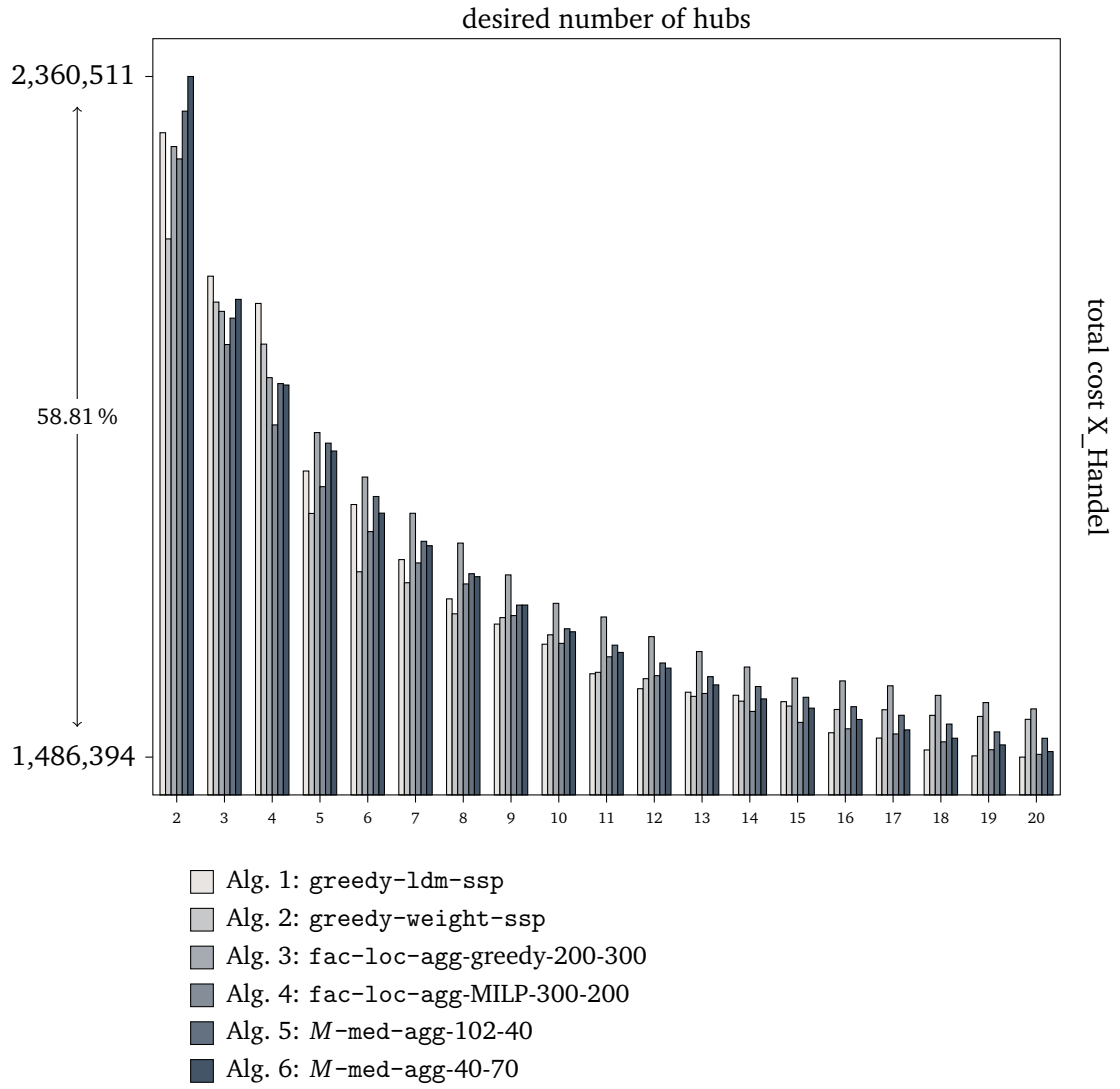


Figure 5.14: Evaluation of aggregation techniques on X_Handel, for detailed numbers see Table A.20.

Table 5.11: Comparison of running time and average solution quality of aggregation techniques on X_Handel

		Alg. 1	Alg. 2	Alg. 3	Alg. 4	Alg. 5	Alg. 6
running time	[min]	4635.3	4693.2	971.9	1138.9	112.7	111.3
maxGap	[%]	8.15	5.42	7.05	4.78	7.63	9.69
avgGap	[%]	1.75	1.40	4.39	1.00	2.73	2.26

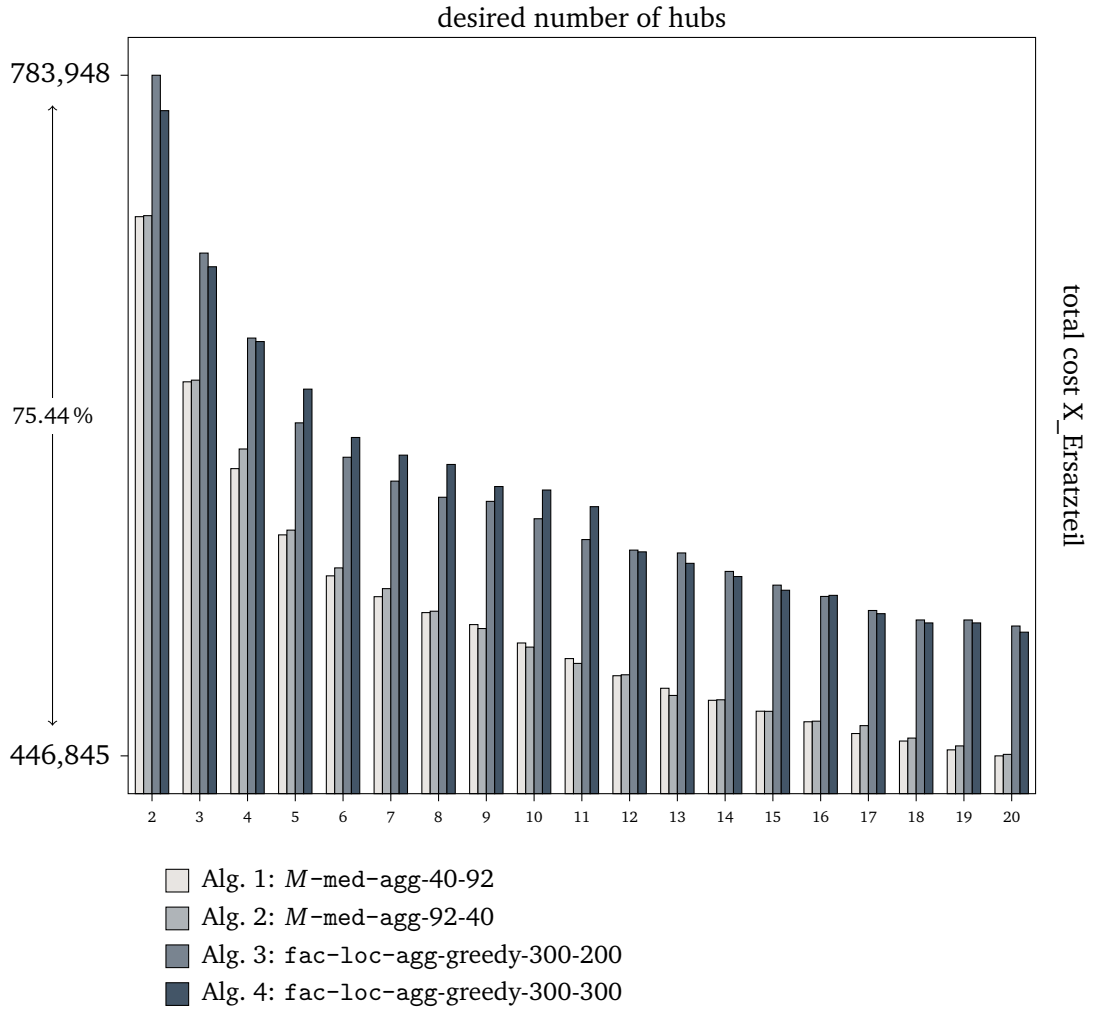


Figure 5.15: Evaluation of aggregation techniques on X_Ersatzteil, for detailed numbers see Table A.21.

Table 5.12: Comparison of running time and average solution quality of aggregation techniques on X_Ersatzteil

		Alg. 1	Alg. 2	Alg. 3	Alg. 4
running time	[min]	136.7	101.5	4831.6	7386.1
maxGap	[%]	0.75	1.65	14.80	15.76
avgGap	[%]	0.11	0.31	12.33	12.87

5.4.8 Summary

We tested the MILP-based heuristic Algorithm 5.2 as the only algorithm that finds solutions to M-HUBROUTE directly, at least for the small instances H_Auto_1 and H_Auto_4. This allows a comparison with incremental solutions that are obtained by a decremental framework on these instances. We would expect that a non-incremental solution for the problem M-HUBROUTE would beat an incremental solutions at the hub bound M with respect to solution quality. This true for for H_Auto_4 with few exceptions, but for H_Auto_1 the variant `cost-close` is slightly better on average than Algorithm 5.2. Since both algorithms are heuristic, we assume the usage of the exact cost function of `cost-close` to be the main advantage over Algorithm 5.2 here.

More importantly, for these two instances we cannot observe a very high price of being incremental. This is of course a weak statement as Algorithm 5.2 could be a weak heuristic failing at finding good non-incremental solutions while the true price of being incremental is still very high. We asked our project partner 4flow AG [4fl17] for quick shot solutions that a practitioner is able to plan within one day assisted by standard logistics software to understand the price of being incremental better. For H_Auto_1 the solution provided this way uses five hubs, at a cost of 312 022 which is indeed 1.6% better then our solution obtained by Algorithm 5.2 for that hub bound. For H_Auto_4 however the quick shot solution uses three hubs and has a cost of 265 816, thereby being 13% worse than the solution from Algorithm 5.2.

These results suggest that the price of being incremental is low in general and thus we tested several variants of our decremental framework in a broad test set, including pure combinatorial as well as MILP-based algorithms.

The purely combinatorial heuristics that rely on USAGE functions from Subsection 5.3.1 for heuristic decrements are one or sometimes two orders of magnitude faster than the `cost-close` variant using the cost criterion $\Delta(n, \mathcal{R}_h)$. On the one hand, the faster variants show cost jumps for smaller hub bounds that the slower variant seems to avoid, but on the other hand, when applied directly to the larger instances of the test set, the slower variant exceeds our time limits.

As a possible remedy we propose an aggregation of sources and sinks into meta-nodes and test two algorithms that compute such an aggregation for a proof of concept. The first and more general algorithm `fac-loc-agg` shows reasonable results on all instances as long as the degree of aggregation remains moderate, i.e. we keep enough of the original sources or sinks that provide many demands in the aggregated instance. Moreover, when combined with `cost-close` it yields the best results for X_Handel over all tested algorithm. This instance is also the largest one, for which a quick shot solution is provided by 4flow AG [4fl17]: This solution uses 15 hubs at a cost of 1 950 398 which is 27 % worse than the best solutions we could compute for that bound using aggregation techniques.

Our second algorithm for computing a meta-node aggregation is M`-med-agg`, which is only suited for instances where the single sourcing assumption applies. This only seems to be the case for the instances X_Handel and X_Ersatzteil in our test set. On X_Ersatzteil it is on average 12% better than the `fac-loc-agg` variants, which could only be applied in rather weak configurations due to the large instance size.

We also mention that we were not provided with a quick shot solution because at the time of this research an optimization over X_Handel was not possible with standard software.

These last results indicate that further research is needed to better understand and correct errors that are introduced by the aggregation techniques presented so far.

5.5 Cost Robust Hub Location

We are interested in the concept of cost robustness that was discussed in Chapter 4 for strategic route planning. The solution strategies therein rely on MILP techniques and we have already seen in the computational study from Section 5.4 that the MILP-based heuristics for hub location can be successfully applied to the medium sized test instances H_Auto_1 and H_Auto_4. It is thus a natural question whether we can combine the concept of cost robustness with our models and heuristics for hub location.

5.5.1 LP-based Search for Cost Robust Hub Location

We start with modifications of our solvable model for hub location towards a solvable model for cost robust hub location. To achieve this, we reapply the modelling techniques from Section 4.6 to deduce a solvable model for M-HUBROBROUTE from the definition in (5.7). We denote the resulting model with ROB-HUBLINPATH.

Our instances for a cost robust hub location have many potential hub nodes such that it is necessary to aggregate demands based on common source and sink nodes as described in Subsection 4.7.5. As a consequence we also work with the relaxed uncertainty set

$$\{(\varrho_j)_{j\in\bar{J}} \;\big|\; \varrho_j \in B_j(G_j), \sum_{j\in\bar{J},\, \varrho_j\neq\bar{\varrho}_j} G_j \leq \Gamma\}, \tag{4.55 rev}$$

which relies on an affine sets B_j for deviations of property vectors

$$B_j(G_j) := \{\bar{\varrho}_j + \delta\tilde{\varrho}_j \;\big|\; \delta \in \{0,1\}\}. \tag{4.53 rev}$$

However, we alter the definition for deviations $\tilde{\varrho}_{j\pi}$ of aggregated properties from the one in Subsection 4.7.6: Instead of computing the values $\tilde{\varrho}_{j\pi}$ from observed deviations of original demands in historic data, we now observe the deviations $\tilde{\varrho}_{j\pi}$ for aggregated demands directly. We give further details on how we observe the values $\tilde{\varrho}_{j\pi}$ in Subsection 5.5.2. For brevity of notation we use $\eta_{kj\pi} := \bar{\varrho}_{j\pi}/\beta_{\pi k}$ and $\tilde{\eta}_{kj\pi} := \tilde{\varrho}_{j\pi}/\beta_{\pi k}$ for the properties and property deviations of demands throughout this section.

Starting with the solvable model HUBLINPATH we need to linearize the exact adversary ADV($\mathcal{R}$) to integrate it. We obtain for a fixed path solution $\mathcal{R} = (r_j^P)_{P\in\mathcal{P},\, j\in J}$ and a fixed facility assignment matrix $Z := (z_{kj})_{k\in K,\, j\in J}$ the adversary problem ADVLIN2($\mathcal{R}, Z$):

ADVLIN2($\mathcal{R}, Z$) $=$

$$\max \quad \sum_{k\in K} m_k \left(\sum_{\pi\in\Pi}\sum_{j\in J} \tilde{\sigma}^*_{kj\pi}\chi_{kj\pi} - \sum_{\pi\in\Pi} s_{k\pi}\theta_{k\pi} \right) + \sum_{j\in J}\sum_{P\in\mathcal{P}_j} (c_j^P + \tilde{c}_j^P\mu_j) r_j^P \tag{5.55}$$

$$\text{s.t. :} \qquad \sum_{j\in J} \mu_j \;\leq\; \Gamma \qquad [\omega \geq 0]$$

(4.34 rev)

$$\mu_j \leq 1 \qquad \forall\, j \in J \qquad [\delta_j \geq 0] \tag{4.38 rev}$$

$$\sum_{\pi\in\Pi} \theta_{k\pi} \leq 1 \qquad \forall\, k \in K \qquad [\phi_k \geq 0] \tag{4.35 rev}$$

$$\sum_{\pi\in\Pi} \chi_{kj\pi} \leq \mu_j \qquad \forall\, k \in K,\, j \in J \qquad [\gamma_{kj} \geq 0] \tag{4.39 rev}$$

$$\chi_{kj\pi} \leq \theta_{k\pi} \qquad \forall\, k \in K,\, j \in J,\, \pi \in \Pi \qquad [\xi_{kj\pi} \geq 0] \tag{4.40 rev}$$

$$\mu_j,\, \theta_{k\pi},\, \chi_{kj\pi} \geq 0 \qquad \forall\, k \in K,\, \pi \in \Pi,\, j \in J \tag{5.56}$$

Here we set $\tilde{\sigma}^*_{kj\pi} := \tilde{\eta}_{kj\pi} z_{kj}$ for shorter notation. Now we can integrate ADVLIN2 via its dual into HUBLINPATH and obtain ROB-HUBLINPATH. In Appendix B.2 we give the full formulations ROB-HUBLINPATH and its dual ROB-HUBLINPATHDUAL with the notion of observed deviations $\tilde{\varrho}_{j\pi}$, along with all other models of this section.

If we now try to apply the solution techniques from Section 5.2 to the cost robust model ROB-HUBLINPATH, then, as a first step, we obtain a violation LP denoted with ROB-VIOLATION via the dual ROB-HUBLINPATHDUAL. The full formulation is given in Appendix B.2. Its structure is similar to VIOLATION from Subsection 5.2.2. For example Inequality (5.14) has a correspondence with Inequality (5.57) in ROB-VIOLATION:

$$RZ_{e(k)j} \leq \sum_{\pi\in\Pi} \eta_{kj\pi} ZL_{k\pi} + ZY_{kj} + \sum_{\pi\in\Pi} m_k \tilde{\eta}_{kj\pi} \chi_{kj\pi}$$
$$\forall j,\, \forall e \in E \setminus \big((F_j) \times (F_j)\big),\, \forall k \in K(e). \tag{5.57}$$

Here, the variables $\chi_{kj\pi}$ decide for the adversary, whether demand j has a deviating property π on tariff level k in the current worst case scenario, and if so, this can increase the cost share $RZ_{e(k)j}$ on the underlying edge e. ROB-VIOLATION also contains some of the Inequalities (4.35, 4.39, 4.40) that model the linearized adversary problem ADVLIN2 before dualization: For example, variables $\chi_{kj\pi}$ are bound by fixed variables $\bar{\mu}_j$ that decide whether some demand j is selected in the worst case scenario of the current restricted master problem. Similar for handling costs, a deviating counterpart $\widetilde{hand}_j(e)$ is incurred by demand j on edge e, if it is in the current worst case scenario. Thus $\widetilde{hand}_j(e)\bar{\mu}_j$ is added to Inequalities (5.30–5.32).

A next step could be to adjust our heuristics from Section 5.2 for application to ROB-VIOLATION and investigate corresponding variants of Theorem 5.3 and Lemma 5.4. However, for Lemma 5.4 we obtain a counterexample from Example 4.1. It exploits the modeling error introduced by relaxing the adversary problem towards ADVLIN2: Indeed, when adapting the example properly to the full formulation ROB-HUBLINPATHDUAL, then the fractional choices $\theta_{11} = \theta_{12} = 0.5$ for solution $(\mathcal{R}_1, Z_1)$ imply fractional values $ZL_{11} = ZL_{12} = 0.5$ with Inequalities (5.17) and (B.4). One can also assert that in this instance any optimum of ADVLIN2 requires fractional values $\theta_{11} = \theta_{12} = 0.5$ which means that the values $ZL_{1\cdot} \in \{0,1\}^\Pi$, as required by Lemma 5.4, are not optimal for this instance.

Thus proving a variant of Theorem 5.3 for ROB-VIOLATION may be technically more involved, but we conjecture that it is still possible. On the other hand, from the heuristic

context of Section 5.1.1, we do not aim at solving ROB-VIOLATION to optimality. Instead we heuristically anticipate the variables $\chi_{kj\pi}$. We suggest the fixation $\theta_{k\pi} = ZL_{k\pi}/m_k$ for each facility k and $\chi_{kj\pi} = \min\{\bar{\mu}_j, \theta_{k\pi}\}$ for each k, j, π. This is a heuristic restriction that may increase optimum solution cost. Including all fixations in ROB-VIOLATION actually turns it into a version of VIOLATION in which $\bar{\varrho}_j + \bar{\mu}_j\tilde{\varrho}_j$ is the property vector for demand j. These fixations enable us to apply our heuristic framework Algorithm 5.2 from Section 5.2 also for the robust setting. On the downside, with these fixations we loose the certificate of optimality from solving ROB-VIOLATION: Since the fixations tighten the right hand side of Inequalities 5.57, they lead to unnecessary high values ZY_{kj}. So if an optimum of an restricted master problem of ROB-HUBLINPATH exploits the modeling error of ADVLIN2, then we could still find an optimum of the corresponding ROB-VIOLATION with nonzero violation. Thus Theorem 5.3 cannot be established for ROB-VIOLATION with these fixations.

5.5.2 Trend Based Uncertainty Sets from Observing Aggregated Demands

Our methods require the input data to be given as uncertainty intervals $[\bar{\varrho}_{j\pi}, \hat{\varrho}_{j\pi}]$ for each super demand j and each property π. For our case studies we are provided with weekly demand values for each original demand over several weeks. We use this data to derive interval data for the properties of aggregated demands.

As we are not aware of any widely accepted method for this task, we present a method of common sense here that does not rely on particular insights on the economics of the underlying networks. It is well suited to show the capabilities of robust optimization while its application in practice still needs to be evaluated.

The Outlier Method Given a discrete set T of observations for original demand values we can compute property vectors $\varrho_j^t \in \mathbb{R}^\Pi$ for each observation $t \in T$ and each aggregated demand j. The *outlier method* considers some of the observations $\varrho_{j\pi}^t$ as outliers so as to reflect a rough trend of a change in demand values that may be known or assumed for the whole network. The uncertainty sets that we propose can be characterized by two parameters:

- **"LB_x"**: For some value x we let $\bar{\varrho}_{j\pi}$ be the $(x+1)$-lowest nonzero value in $(\varrho_{j\pi}^t)_{t\in T}$, that is, we consider the x lowest values as outliers. Further we denote with "LB_avg" the setting in which we chose $\bar{\varrho}_{j\pi}$ as the average value of $(\varrho_{j\pi}^t)_{t\in T}$. We refer to this setting as the *artificial average case szenario* and to solutions that are optimized for this scenario as *deterministic solutions*.

- **"UB_y"**: Similarly, for some y we let $\hat{\varrho}_{j\pi}$ be the $(y+1)$-highest value in $(\varrho_{j\pi}^t)_{t\in T}$, thereby considering the y highest values as outliers.

We then denote a specific uncertainty set obtained by the outlier method with LB_x-UB_y.

5.6 Computational Results for Cost Robust Hub Location

In this section we evaluate Algorithm 5.2 for the cost robust model ROB-HUBLINPATH for aggregated demands. For the aggregation, we let each aggregated demand j consume a robustness budget G_j equal to $\tilde{\varrho}_{j\pi^*}$ for some designated property π^*; in the tests this property

will be "loading meters". We optimize and evaluate for robust worst case scenarios for each uncertainty budget Γ in $\{0\%, 5\%, 10\%, 20\%, 40\%\}$ of $\sum_{j\in\bar{J}} \bar{\varrho}_{j\pi^*}$. This allows the interpretation of Γ to be the percentage of overall nominal transported volume of property π^*.

Table 5.13: Tuning parameters for the subroutines of in Algorithm 5.2

parameter	setting	explanation
k_2	6	how many outgoing (respectively incoming) edges to select per hub node, per demand in hopNewHubs
k_4	5	how many node neighbors to add in hopNewHubs
k_7	5	threshold for selectPaths, for how many hubs to add paths
k_8	6	how many paths to add for each demand in a starting formulation, see Section 5.2.7
k_{11}	7	how many pricing rounds to perform in Line 2 of Algorithm 5.2

The LP-model ROB-HUBLINPATH involves considerably more variables and constraints than the deterministic model HUBLINPATH. We use two measures to achieve that Algorithm 5.2 terminates within a time limit of two hours for the smaller instances H_Auto_1 and H_Auto_4 described in Subsection 5.4.1: First, we preselect 15 potential hub nodes using the decremental configuration `greedy-weight-ssp` with the deterministic model. The configuration details are specified in Table 5.3. This bound seems large enough to include all relevant hub nodes for a desired bound of six hubs. Second, we tune the parameters of Algorithm 5.2 according to Table 5.13 where all parameters not mentioned are kept as specified in Table 5.1 from Subsection 5.2.8.

We aim at selecting six of the remaining 15 hubs with minimum robust cost: Following the suggestion of our project partner 4flow AG [4fl17], this represents a reasonable and interesting hub bound for both instances. We use trend based uncertainty sets as described in Subsection 5.5.2 and generate all uncertainty sets LB_x-UB_y for (x, y) in $\{\text{avg}, 1, 4\}\times\{0, 1, 4\}$. For each uncertainty set we use Algorithm 5.2 to compute cost robust solutions for each Γ in $\{0\%, 5\%, 10\%, 20\%, 40\%\}$. In our figures and tables we use the letter g instead of Γ, e.g. $g = 5\%$ refers to $\Gamma = 5\%$.

Our method of evaluation is very similar to the one in Subsection 4.8.1. We use the exact adversary problem ADV for aggregated demands from (5.4), Section 5.1.1 to evaluate each solution with respect to their specific worst case scenarios for Γ in $\{0\%, 5\%, 10\%, 20\%, 40\%\}$.

Furthermore for each solution we evaluate the *average historical cost*, that is, the cost incurred by observed demand values ϱ_j^t and averaged over all observations $t \in T$. Since we have not presented a method to optimize for average historical cost directly, we optimize for the artificial average scenario instead. This typically gives good solutions. Altogether we obtain six solutions for each uncertainty set LB_x-UB_y and for each solution we report six evaluation criteria.

The results show similarities across the uncertainty sets and we focus on the two uncertainty sets LB_avg-UB_1 and LB_1-UB_4. The cost of solutions for all criteria are summarized in Tables 5.14 and 5.15 for H_Auto_1, and in Tables 5.16 and 5.17 for H_Auto_4. For results of all other uncertainty sets we refer to Appendix A.2.2 for H_Auto_1 and Appendix A.2.2 for H_Auto_4.

Each row in our tables refers to a fixed solution and each column to the cost of this

solution for one of the different criteria. The first column specifies how the solution was obtained. Here, the first line "opt. for avg. scen." is the best known solution for the artificial average case scenario for this hub bound. All other lines refer to robust solutions optimized for worst case cost. In each column we mark the best entry for this criterion in bold and give the relative percentage deviation from the best value in parenthesis.

5.6.1 Robust and Average Historical Cost

We first focus on the uncertainty sets that contain the artificial average case scenario as the lower bounds of all uncertainty intervals. This implies that the solution cost for $\Gamma = 0\%$ compares with the deterministic solutions from Section 5.4. Also the robust solution for $\Gamma = 0\%$ has in this case indeed been optimized for the average case scenario. In this regard, as a heuristic for the deterministic case, the algorithms proposed for robust optimization are weaker than the heuristics from Section 5.4. Clearly, the linearizations in order to model tariff cost with linear programming introduce some inaccuracies and also the algorithmic set up from Table 5.13 is tailored for robust optimization. With a focus on LB_avg-UB_1 we observe for H_Auto_1 that the (robust) solution for $\Gamma = 0\%$ to be 1% worse for the average case scenario than the best known deterministic solution, as shown in Table 5.14, and for H_Auto_4 we have even a 2.4% worse solution, see Table 5.16.

For the observed average historical cost the same gap is preserved for H_Auto_1 while it reduces to 0.5% for H_Auto_4. Understanding that a price of robustness is to be measured in terms of observed historical costs, and not for the artificial average case scenario, this price starts at 1% for less conservative solutions, i.e. those with a small value Γ, and raises to 2% for more conservative solutions. The benefit for robust cost compared to the best known solution for the average case scenario rises to 2.5% on H_Auto_1 and even to 4.3% on H_Auto_4. Solutions that achieve a good compromise between robust and observed historical cost may be found for $\Gamma = 5\%$ or $\Gamma = 10\%$.

5.6.2 Trend Based Uncertainty Sets

Uncertainty sets that do not have the artificial average case demand values as lower bounds of uncertainty intervals, have significantly higher average historical cost, up to 4.1% more than respective cost of the best known deterministic solution, e.g. in Table 5.17 for the uncertainty set LB_1-UB_4 and $\Gamma = 10\%$. If a future realization of demand values, that follows similar patterns as observed historically, is likely, then the interval lower bounds should be chosen as the artificial average case scenario. If the artificial average case scenario should be avoided—it is artificial—then small uncertainty intervals, as for example in the uncertainty sets LB_4-UB_4, see Tables A.30 and A.39, lead to better average historical cost than large uncertainty intervals, e.g. LB_1-UB_0 in Tables A.25 and A.34. For LB_4-UB_4 the gap to the best known solution is close to 1% on H_Auto_1 and between 2% and 3% on H_Auto_4.

In situations where a repetition of historic demand values is considered unlikely it can be worthwhile to immunize against demand values that realize an anticipated trend. As an example we focus on the set LB_1-UB_4: Here the cost for the interval lower bounds, i.e. $\Gamma = 0\%$ are below the observed historical average costs and the worst cases for $\Gamma = 5\%$ are similar to them, see Tables 5.17 and 5.15. Thus the underlying trend could be subsumed as

"fluctuations with a moderate increase of demand values". The worst case scenarios for the set LB_1-UB_4 have considerably lower costs when comparing to worst case scenarios from the uncertainty set LB_avg-UB_1 for the corresponding values for Γ.

Surprisingly, the best known deterministic solution has a very large gap for the worst case cost for $\Gamma = 5\%$, that is, up to 5.3% for H_Auto_1 and even up to 10.8% on H_Auto_4. This is especially interesting, since on the one hand, the deterministic solution has been optimized for a different scenario, i.e. the artificial average case scenario, but on the other hand the worst case cost for $\Gamma = 5\%$ is close to the historical average cost and is therefore not over-pessimistic.

Also the robust solutions show rather moderate gaps on the worst case costs for the value of Γ that they are not optimized for. We conclude that an immunization is also possible for these trend based uncertainty sets, but they have a higher price of robustness than for the LB_avg-UB_1 uncertainty sets.

5.7 Conclusion

This chapter presents an algorithmic toolkit to obtain robust solutions to the M-median hub location problem. We address two concepts for robustness: First, incremental hub chains are composed of solutions that are robust under a changing number of hub nodes to operate. Second, cost robust solutions are robust for worst-case cost under uncertainty for demand values.

For incremental hub chains we propose a flexible decremental framework that can be applied to very large instances in our test set when combined with fast combinatorial heuristics. To asses the price of being incremental we compare incremental solutions with M-median solutions that are obtained with an LP-based hub search on two medium sized instances. Surprisingly, the price of being incremental observed is almost negligible. However, a hard statement for a price of being incremental is subject to further research.

We applied LP-based hub search to a robust model to obtain cost robust solutions for medium size instances using uncertainty sets that show the capabilities of robust optimization. An a posteriori evaluation with exact worst cases shows a price of robustness that is moderate, when the uncertainty set contains the artificial average case scenario as lower bounds: Roughly one percent loss for historical average costs is compensated by two or four percent gain of robust solutions for their specific worst case costs. For trend based uncertainty sets that do not contain the artificial average case as interval lower bounds, this tradeoff increases: Here, three percent loss in historical average cost can save up to ten percent for worst case costs.

For larger instances, even the combinatorial algorithms show very long running times. Identifying the large number of demands as bottleneck, we present a generic aggregation concept and test two aggregation techniques. This way even, for the largest instance we obtain incremental hub chains, which to the best of our knowledge has not been achieved for instances of this size before. Fluctuations in solution quality hint that further improvements may be possible.

5.7.1 Outlook

This chapter leaves several interesting question for further research. Starting with LP-based techniques, we have established an optimality criterion based on a dual restricted master

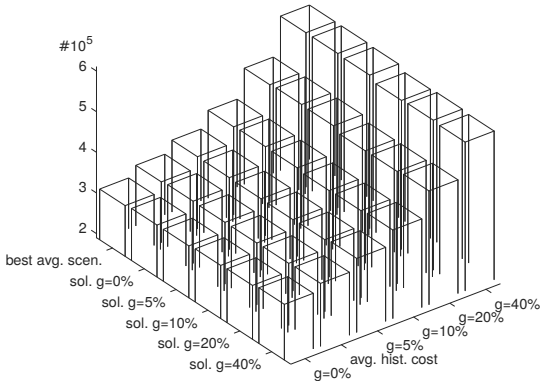
problem. Unfortunately, we cannot solve this dual, as it contains as many variables there are edges times demands, i.e. for the tuples in $E \times J$. Growing the dual only by one hop path alternatives turned out to sufficient for logistics network, as they require only short paths in final solutions. When applied to other hub location problems there may be adjustments required.

On the other hand, feasible node potentials B_{jn} for the dual master problem could be sufficient to show optimality. Note that we can indeed efficiently check their feasibility as this problem decomposes on transportation relations and leaves only a very small LP to check. It would be interesting whether a fast combinatorial algorithm can be found, that constructs feasible node potentials, if they exist, and that is able to prove feasibly extensibility of the current dual solution in the dual master problem.

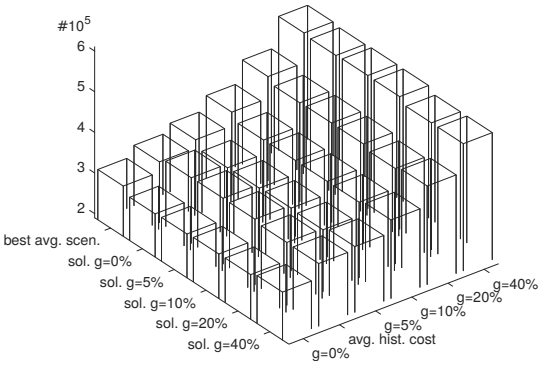
Furthermore we consider the LP-based hub search a dual heuristic that offers a global perspective on hub location. However, it is limited by memory requirements and degeneracy issues of simplex based algorithms, putting aside license restrictions when implemented in commercial software. From a practical perspective it can be worthwhile to capture the features of the LP-based hub search in a purely combinatorial heuristic.

Improvements for our aggregation techniques could also be possible: A major drawback is that decisions for the aggregation phase remain fixed during the optimization phase. Consider for example decremental algorithms: It is obvious that aggregation decisions can be resolved more accurately if already some of the hubs have been closed. Thus, we need dynamic data structures that allow adjustments of the aggregation during the optimization phase.

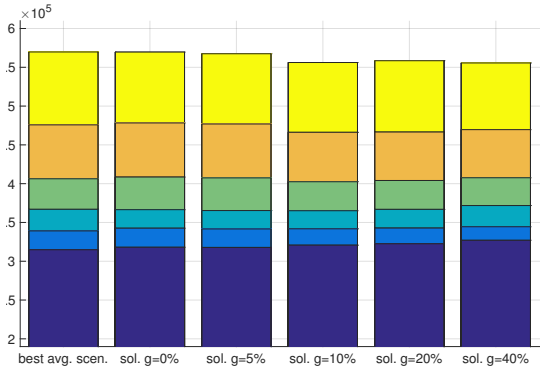
Finally, our cost robust optimization can be directly applied to an aggregated instance. This means that many demands of a meta-node may appear as one demand for robust optimization, and consequently their property deviations are observed as an aggregation of original demands. Thus a specific aggregation influences the uncertainty sets that are considered, which is not desired. It would be interesting to investigate such dependencies and methods to overcome them.



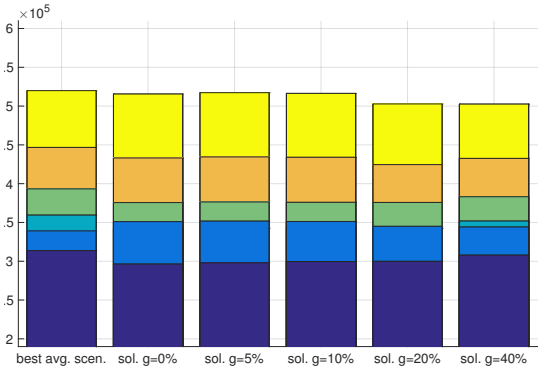
(a) Robust costs on LB_avg-UB_1



(b) Robust costs on LB_1-UB_4



(c) Front view for LB_avg-UB_1



(d) Front view for LB_1-UB_4

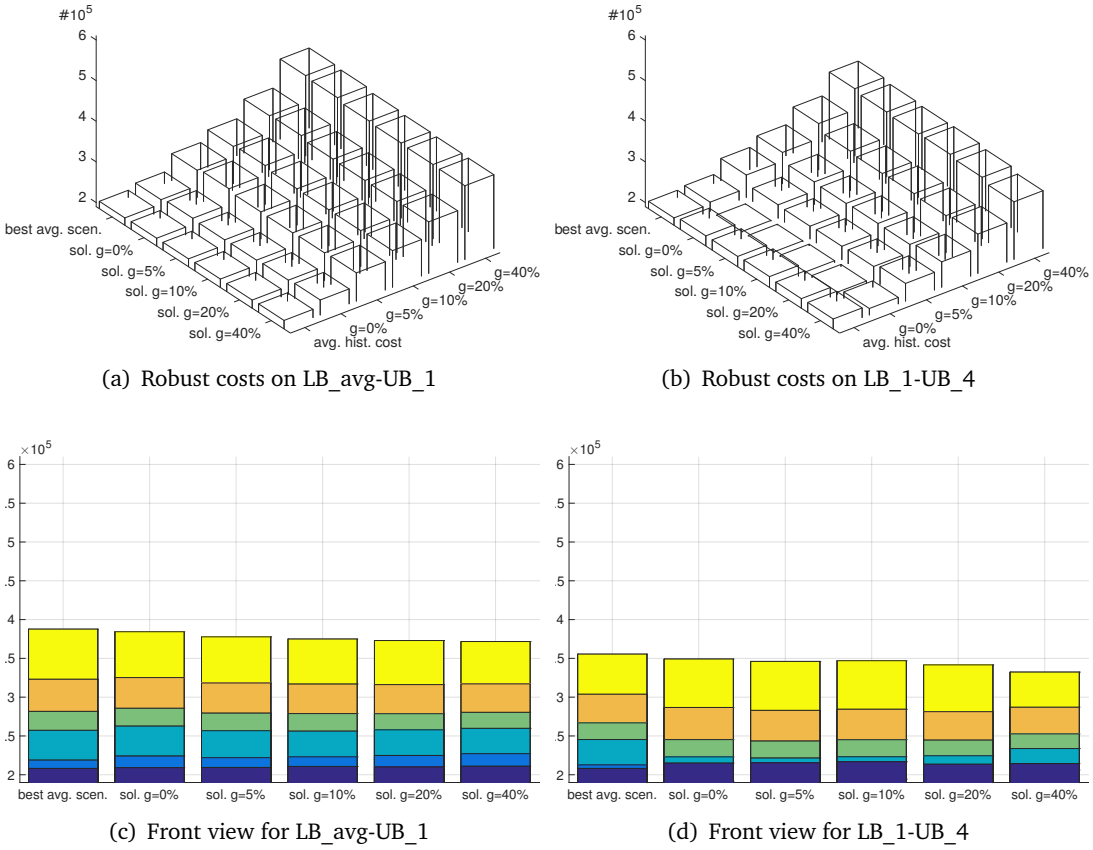
Figure 5.16: Evaluation of robust solutions for H_Auto_1 on uncertainty sets LB_avg-UB_1 and LB_1-UB_4

Table 5.14: Detailed costs of robust solutions of H_Auto_1 LB_avg-UB_1 (left column)

opt. for	avg hist. cost	wc-sc. g=0%	wc-sc. g=5%	wc-sc. g=10%	wc-sc. g=20%	wc-sc. g=40%
avg. scen.	**339 114** (0.0)	**314 810** (0.0)	366 935 (0.5)	406 288 (0.9)	475 779 (2.1)	569 739 (2.5)
g=0%	342 606 (1.0)	318 052 (1.0)	366 474 (0.4)	408 528 (1.5)	478 257 (2.6)	569 599 (2.5)
g=5%	341 624 (0.7)	317 801 (1.0)	365 331 (0.1)	407 435 (1.2)	476 910 (2.3)	567 458 (2.1)
g=10%	341 813 (0.8)	320 757 (1.9)	**365 112** (0.0)	**402 476** (0.0)	**466 198** (0.0)	556 162 (0.1)
g=20%	342 781 (1.1)	322 531 (2.5)	366 866 (0.5)	404 096 (0.4)	466 549 (0.1)	558 469 (0.5)
g=40%	344 457 (1.6)	327 026 (3.9)	371 784 (1.8)	407 615 (1.3)	469 628 (0.7)	**555 585** (0.0)

Table 5.15: Detailed costs of robust solutions of H_Auto_1 LB_1-UB_4 (right column)

opt. for	avg hist. cost	wc-sc. g=0%	wc-sc. g=5%	wc-sc. g=10%	wc-sc. g=20%	wc-sc. g=40%
avg. scen.	**339 114** (0.0)	313 751 (5.9)	359 440 (5.3)	393 265 (4.7)	446 689 (5.2)	519 914 (3.5)
g=0%	351 079 (3.5)	**296 388** (0.0)	**341 415** (0.0)	**375 610** (0.0)	433 119 (2.0)	515 541 (2.6)
g=5%	351 693 (3.7)	297 882 (0.5)	342 489 (0.3)	376 413 (0.2)	434 458 (2.3)	517 281 (2.9)
g=10%	351 259 (3.6)	299 664 (1.1)	344 287 (0.8)	375 922 (0.1)	433 967 (2.2)	516 347 (2.7)
g=20%	345 006 (1.7)	299 886 (1.2)	342 063 (0.2)	375 785 (0.0)	**424 555** (0.0)	502 772 (0.0)
g=40%	344 114 (1.5)	307 899 (3.9)	351 802 (3.0)	383 144 (2.0)	432 548 (1.9)	**502 537** (0.0)



(a) Robust costs on LB_avg-UB_1

(b) Robust costs on LB_1-UB_4

(c) Front view for LB_avg-UB_1

(d) Front view for LB_1-UB_4

Figure 5.17: Evaluation of robust solutions for H_Auto_4 on uncertainty sets LB_avg-UB_1 and LB_1-UB_4

Table 5.16: Detailed costs of robust solutions of H_Auto_4 LB_avg-UB_1 (left column)

opt. for	avg hist. cost	wc-sc. g=0%	wc-sc. g=5%	wc-sc. g=10%	wc-sc. g=20%	wc-sc. g=40%
avg. scen.	**208 120** (0.0)	**218 846** (0.0)	257 202 (0.4)	281 634 (1.1)	323 039 (2.2)	387 651 (4.3)
g=0%	209 155 (0.5)	224 110 (2.4)	262 654 (2.5)	285 638 (2.5)	325 242 (2.9)	384 335 (3.4)
g=5%	209 382 (0.6)	222 014 (1.4)	256 763 (0.2)	279 368 (0.3)	318 272 (0.7)	377 756 (1.6)
g=10%	210 679 (1.2)	223 099 (1.9)	**256 174** (0.0)	278 832 (0.1)	316 859 (0.2)	374 891 (0.9)
g=20%	210 043 (0.9)	224 846 (2.7)	257 883 (0.7)	**278 658** (0.0)	**316 179** (0.0)	372 991 (0.3)
g=40%	210 938 (1.4)	227 142 (3.8)	259 759 (1.4)	280 394 (0.6)	317 099 (0.3)	**371 713** (0.0)

Table 5.17: Detailed costs of robust solutions of H_Auto_4 LB_1-UB_4 (right column)

opt. for	avg hist. cost	wc-sc. g=0%	wc-sc. g=5%	wc-sc. g=10%	wc-sc. g=20%	wc-sc. g=40%
avg. scen.	**208 120** (0.0)	212 706 (11.1)	245 379 (10.8)	266 848 (9.6)	303 731 (8.0)	355 444 (7.0)
g=0%	214 986 (3.3)	**191 511** (0.0)	223 059 (0.7)	245 267 (0.8)	286 704 (1.9)	349 128 (5.1)
g=5%	215 344 (3.5)	192 320 (0.4)	**221 521** (0.0)	**243 416** (0.0)	282 948 (0.6)	346 069 (4.1)
g=10%	216 733 (4.1)	194 365 (1.5)	223 190 (0.8)	245 142 (0.7)	284 385 (1.1)	346 989 (4.4)
g=20%	213 536 (2.6)	195 503 (2.1)	224 464 (1.3)	244 606 (0.5)	**281 296** (0.0)	341 617 (2.8)
g=40%	214 447 (3.0)	206 167 (7.7)	233 757 (5.5)	252 672 (3.8)	286 961 (2.0)	**332 289** (0.0)

Chapter 6

Scheduling Maintenance Jobs in Networks

6.1 Introduction

Transportation and telecommunication networks are important backbones of modern infrastructure and have been a major focus of research in combinatorial optimization and other areas. Research on such networks usually concentrates on optimizing their usage, for example by maximizing throughput or minimizing costs. In the majority of the studied optimization models it is assumed that the network is permanently available, and our choices only consist in deciding which parts of the network to use at each point in time.

Practical transportation and telecommunication networks, however, can generally not be used non-stop. Be it due to wear-and-tear, repairs, or modernizations of the network, there are times when parts of the network are unavailable. We study how to schedule and coordinate such maintenance in different parts of the network to ensure connectivity.

While network problems and scheduling problems individually are fairly well understood, the combination of both areas that results from scheduling network maintenance has only recently received some attention [BKK15a; Bol+14; Nur+12; BKD13; Fla+10] and is theoretically hardly understood.

Problem Definition In this chapter we study connectivity problems which are fundamental in this context. In these problems, we aim to schedule the maintenance of edges in a network in such a way as to preserve connectivity between two designated vertices. Given a network and maintenance jobs with processing times and feasible time windows, we need to decide on the temporal allocation of the maintenance jobs. While a maintenance on an edge is performed, the edge is not available. We distinguish between MINCONNECTIVITY, the problem in which we minimize the total time in which the network is disconnected, and MAXCONNECTIVITY, the problem in which we maximize the total time in which it is connected.

In both of these problems, we are given an undirected graph $G = (V, E)$ with two distinguished vertices $s^+, s^- \in V$. We assume w. l. o. g. that the graph is simple; we can replace a parallel edge $\{u, w\}$ by a new node v and two edges $\{u, v\}, \{v, w\}$. Every edge $e \in E$ needs to undergo $p_e \in \mathbb{Z}_{\geq 0}$ time units of maintenance within the time window $[r_e, d_e]$ with $r_e, d_e \in \mathbb{Z}_{\geq 0}$,

where r_e is called the release date and d_e is called the deadline of the maintenance job for edge e. An edge $e = \{u, v\} \in E$ that is maintained at time t, is not available at t in the graph G. We consider preemptive and non-preemptive maintenance jobs. If a job must be scheduled non-preemptively then, once it is started, it must run until completion without any interruption. If a job is allowed to be preempted, then its processing can be interrupted at any time and may resume at any later time without incurring extra cost.

A *schedule* S for G assigns the maintenance job of every edge $e \in E$ to a single time interval (if non-preemptive) or a set of disjoint time intervals $S(e) := \{[a_1, b_1], \ldots, [a_k, b_k]\}$ (if preemptive) with

$$r_e \leq a_i \leq b_i \leq d_e, \text{ for } i \in [k] \text{ and } \sum_{[a,b] \in S(e)} (b - a) = p_e.$$

If not specified differently, we define $T := \max_{e \in E} d_e$ as our *time horizon*. We do not limit the number of simultaneously maintained edges.

For a given maintenance schedule, we say that the network G is *disconnected at time t* if there is no path from s^+ to s^- in G at time t, otherwise we call the network G *connected at time t*. The goal is to find a maintenance schedule for the network G so that the total time where G is disconnected is minimized (MinConnectivity). We also study the maximization variant of the problem, in which we want to find a schedule that maximizes the total time where G is connected (MaxConnectivity).

Our Results. For *preemptive* maintenance jobs, we show that we can solve both problems, MaxConnectivity and MinConnectivity, efficiently in arbitrary networks (Theorem 6.1). This result crucially requires that we are free to preempt jobs at arbitrary points in time. Under the restriction that we can *preempt* jobs only at *integral points in time*, the problem becomes $\mathcal{NP}$-hard. More specifically, MaxConnectivity does not admit a $(2 - \epsilon)$-approximation algorithm for any $\epsilon > 0$ in this case, and MinConnectivity is inapproximable (Theorem 6.4), unless $\mathcal{P} = \mathcal{NP}$. By inapproximable, we mean that it is $\mathcal{NP}$-complete to decide whether the optimal objective value is zero or positive, leading to unbounded approximation factors.

This is true even for unit-size jobs. This complexity result is interesting and may be surprising, as it is in contrast to results for standard scheduling problems, without an underlying network. Here, the restriction to integral preemption typically does not increase the problem complexity when all other input parameters are integral. However, the same question remains open in a related problem concerning the busy-time in scheduling, studied in [CKM15; CKM14].

For *non-preemptive* instances, we establish that there is no $(c\sqrt[3]{|E|})$-approximation algorithm for MaxConnectivity for some constant $c > 0$ and that MinConnectivity is inapproximable even on disjoint paths between two nodes s and t, unless $\mathcal{P} = \mathcal{NP}$ (Theorems 6.5,6.6). On the positive side, we provide an $(\ell + 1)$-approximation algorithm for MaxConnectivity in general graphs (Theorem 6.8), where ℓ is the number of distinct latest start times (deadline minus processing time) for jobs.

We use the notion *power of preemption* to capture the benefit of allowing arbitrary job preemption. The power of preemption is a commonly used measure for the impact of preemption in scheduling [CI98; CSV12; SM02; SS14]. Other terms used in this context include *price of non-preemption* [Coh+15], *benefit of preemption* [PS95] and *gain of preemption* [Ha92]. It

is defined as the maximum ratio of the objective values of an optimal non-preemptive and an optimal preemptive solution. We show that the power of preemption is $\Theta(\log |E|)$ for MINCONNECTIVITY on a path (Theorem 6.9) and unbounded for MAXCONNECTIVITY on a path (Theorem 6.12). This is in contrast to other scheduling problems, where the power of preemption is constant, e. g. [CSV12; SM02].

On paths, we show that *mixed* instances, which have both preemptive and non-preemptive jobs, are weakly $\mathcal{NP}$-hard (Theorem 6.13). This hardness result is of particular interest, as both purely non-preemptive and purely preemptive instances can be solved efficiently on a path (see Theorem 6.1 and [Kha+15]). Furthermore, we give a simple 2-approximation algorithm for mixed instances of MINCONNECTIVITY (Theorem 6.14).

Related Work. The concept of combining scheduling with network problems has been considered by different communities lately. However, the specific problem of only maintaining connectivity over time between two designated nodes has not been studied to our knowledge. Boland et al. [BKK15a; BKK15b; Bol+14] study the combination of non-preemptive arc maintenance in a transport network, motivated by annual maintenance planning for the Hunter Valley Coal Chain [BS12]. Their goal is to schedule maintenance such that the maximum s-t-flow over time in the network with zero transit times is maximized. They show strong $\mathcal{NP}$-hardness for their problem and describe various heuristics and IP based methods to address it. Also, they show in [BKK15b] that in their non-preemptive setting, if the input is integer, there is always an optimal solution that starts all jobs at integer time points. In [BKK15a], they consider a variant of their problem, where the number of concurrently performable maintenances is bounded by a constant.

Their model generalizes ours in two ways – it has capacities and the objective is to maximize the total flow value. As a consequence of this, their IP-based methods carry over to our setting, but these methods are of course not efficient. Their hardness results do not carry over, since they rely on the capacities and the different objective. However, our hardness results – in particular our approximation hardness results – carry over to their setting, illustrating why their IP-based models are a good approach for some of these problems.

Bley, Karch and D'Andreagiovanni [BKD13] study how to upgrade a telecommunication network to a new technology employing a bounded number of technicians. Their goal is to minimize the total service disruption caused by downtimes. A major difference to our problem is that there is a set of given paths that shall be upgraded and a path can only be used if it is either completely upgraded or not upgraded. They give ILP-based approaches for solving this problem and show strong $\mathcal{NP}$-hardness for a non-constant number of paths by reduction from the linear arrangement problem.

Nurre et al. [Nur+12] consider the problem of restoring arcs in a network after a major disruption, with restoration per time step being bounded by the available work force. Such network design problems over time have also been considered by Kalinowski, Matsypura and Savelsbergh [KMS15].

In scheduling, minimizing the busy time refers to minimizing the amount of time for which a machine is used. Such problems have applications for instance in the context of energy management [Mer+12] or fiber management in optical networks [Fla+10]. They have been studied from the complexity and approximation point of view in [CKM14; Fla+10; Kha+15; Mer+12]. The problem of minimizing the busy time is equivalent to our problem in the case of a path, because there we have connectivity at a time point when no edge in the

path is maintained, i. e., no machine is busy.

Thus, the results of Khandekar et al. [Kha+15] and Chang, Khuller and Mukherjee [CKM14] have direct implications for us. They show that minimizing busy time can be done efficiently for purely non-preemptive and purely preemptive instances, respectively.

6.2 Preemptive Scheduling

In this section, we consider problem instances where all maintenance jobs can be preempted.

Theorem 6.1 *Both* MaxConnectivity *and* MinConnectivity *with preemptive jobs can be solved optimally in polynomial time on arbitrary graphs.*

Proof. We establish a linear program (LP) for MaxConnectivity. Let $TP = \{0\} \cup \{r_e, d_e : e \in E\} = \{t_0, t_1, \dots, t_k\}$ be the set of all *relevant time points* with $t_0 < t_1 < \cdots < t_k$. We define $I_i := [t_{i-1}, t_i]$ and $w_i := |I_i|$ to be the length of interval I_i for $i = 1, \dots, k$.

In our linear program we model connectivity during interval I_i by an (s^+, s^-)-flow $x^{(i)}$, $i \in \{1, \dots, k\}$. To do so, we add for every undirected edge $e = \{u, v\}$ two directed arcs (u, v) and (v, u). Let A be the resulting arc set. With each edge/arc we associate a capacity variable $y_e^{(i)}$, which represents the fraction of availability of edge e in interval I_i. Hence, $1 - y_e^{(i)}$ gives the relative amount of time spent on the maintenance of edge e in I_i. Additionally, the variable f_i expresses the fraction of availability for interval I_i.

$$\max \quad \sum_{i=1}^{k} w_i \cdot f_i \tag{6.1}$$

$$\text{s.t.} \quad \sum_{u:(v,u)\in A} x^{(i)}_{(v,u)} - \sum_{u:(u,v)\in A} x^{(i)}_{(u,v)} = \begin{cases} f_i & \forall\, i \in [k],\ v = s^+, \\ 0 & \forall\, i \in [k],\ v \in V \setminus \{s^+, s^-\}, \\ -f_i & \forall\, i \in [k],\ v = s^-, \end{cases} \tag{6.2}$$

$$\sum_{i:I_i \subseteq [r_e, d_e]} (1 - y_e^{(i)}) w_i \geq p_e \qquad \forall\, e \in E, \tag{6.3}$$

$$x^{(i)}_{(u,v)}, x^{(i)}_{(v,u)} \leq y^{(i)}_{\{u,v\}} \qquad \forall\, i \in [k],\ \{u, v\} \in E, \tag{6.4}$$

$$f_i \leq 1 \qquad \forall\, i \in [k], \tag{6.5}$$

$$x^{(i)}_{(u,v)}, x^{(i)}_{(v,u)}, y^{(i)}_{\{u,v\}} \in [0, 1] \qquad \forall\, i \in [k],\ \{u, v\} \in E. \tag{6.6}$$

Notice that the LP is polynomial in the input size, since $k \leq 2|E|$. We show in Lemma 6.2 that this LP is a relaxation of preemptive MaxConnectivity on general graphs and in Lemma 6.3 that any optimal solution to it can be turned into a feasible schedule with the same objective function value in polynomial time, which proves the claim for MaxConnectivity. For MinConnectivity, notice that any solution that maximizes the time in which s and t are connected also minimizes the time in which s and t are disconnected – thus, we can use the above LP there as well. □

Next, we need to prove the two lemmas that we used in the proof of Theorem 6.1. We begin by showing that the LP is indeed a relaxation of our problem.

Lemma 6.2 *The given LP is a relaxation of preemptive* MAXCONNECTIVITY *on general graphs.*

Proof. Given a feasible maintenance schedule, consider an arbitrary interval I_i, $i \in \{1,\dots,k\}$, and let $[a_1^i, b_1^i] \dot\cup \dots \dot\cup [a_{m_i}^i, b_{m_i}^i] \subseteq I_i$ be all intervals where s^+ and s^- are connected in interval I_i. We set $f_i = \sum_{\ell=1}^{m_i} (b_\ell^i - a_\ell^i)/w_i \leq 1$ and set $y_e^{(i)} \in [0,1]$ to the fraction of time where edge e is not maintained in interval I_i. Note that (6.3) is automatically fulfilled, since we consider a feasible schedule. It is left to construct a feasible flow $x^{(i)}$ for the fixed variables f_i and $y^{(i)}$ for all $i = 1,\dots,k$.

Whenever the given schedule admits connectivity we can send one unit of flow from s^+ to s^- along some directed path in G. Moreover, in intervals where the set of processed edges does not change we can use the same path for sending the flow. Let $[a,b] \subseteq I_i$ be an interval where the set of processed edges does not change and in which we have connectivity. Let $\mathcal{C}_i$ be the collection of all such intervals in I_i. Then, we send a flow $x_{[a,b]}^{(i)}$ from s^+ to s^- along any path of total value $(b-a)/w_i$ using only arcs for which the corresponding edge is not processed in $[a,b]$. The flow $x^{(i)} = \sum_{[a,b]\in\mathcal{C}_i} x_{[a,b]}^{(i)}$, which is a sum of vectors, gives the desired flow. The constructed flow $x^{(i)}$ respects the flow conservation (6.2) and non-negativity constraints (6.6), uses no arc more than the corresponding $y_e^{(i)}$, since flow $x^{(i)}$ is driven by the schedule. □

Lemma 6.3 *Any feasible LP solution can be turned into a feasible maintenance schedule at no loss in the objective function value in polynomial time.*

Proof. Let (x,y,f) be a feasible solution of the given LP. Let $\mathcal{P}^i := (P_1^i,\dots,P_{\lambda_i}^i)$ be a path decomposition [KV12] of the (s^+,s^-)-flow $x^{(i)}$ for an arbitrary interval $I_i := [a_i,b_i]$, $i \in \{1,\dots,k\}$, after deleting all flow from possible circulations. Furthermore, let $x(P_\ell^i)$ be the value of the (s^+,s^-)-flow $x^{(i)}$ sent along the directed path P_ℓ^i. For each arc $a \in A$ we have that $\sum_{\ell\in[\lambda_i]:a\in P_\ell^i} x(P_\ell^i) = x_a^{(i)}$ by the definition of $\mathcal{P}^i$. Hence, we get $\sum_{\ell\in[\lambda_i]} x(P_\ell^i) = f_i \leq 1$ by using (6.5). We now divide the interval I_i into disjoint subintervals to allocate connectivity time for each path in our path decomposition. More precisely, we do *not* maintain any arc (u,v) (resp. edge $\{u,v\}$) contained in P_ℓ^i, $\ell = 1,\dots,\lambda_i$, in the time interval

$$\left[a_i + \sum_{m=1}^{\ell-1} w_i \cdot x(P_m^i),\, a_i + \sum_{m=1}^{\ell} w_i \cdot x(P_m^i) \right] \text{ of length } w_i \cdot x(P_\ell^i). \tag{6.7}$$

Inequality (6.4) and $\sum_{\ell\in[\lambda_i]:a\in P_\ell^i} x(P_\ell^i) = x_a^{(i)}$ thereby ensure that by now the total time where edge e does not undergo maintenance in interval I_i equals at most $w_i \cdot y_e^{(i)}$ time units. By Inequality (6.3), we can thus distribute the processing time of the job for edge e among the remaining slots of all intervals I_i, $i = 1,\dots,k$. For instance, we could greedily process the job for edge e as early as possible in available intervals. Note that arbitrary preemption of the processing is allowed. By construction, we have connectivity on path P_ℓ^i, $\ell = 1,\dots,\lambda_i$, for at least $w_i \cdot x(P_\ell^i)$ time units in interval I_i. Thus, the constructed schedule has total connectivity time of at least $\sum_{i=1}^k w_i \sum_{\ell=1}^{\lambda_i} x(P_\ell^i) = \sum_{i=1}^k w_i \cdot f_i$. Since the path decomposition can be computed in polynomial-time and the resulting number of paths is bounded by the number of edges [KV12], we can obtain the feasible schedule in polynomial-time. □

For unit-size jobs we can simplify the given LP by restricting to the first $|E|$ slots within every interval I_i. This, in turn, allows to consider intervals of unit-size, i.e., we have $w_i = 1$ for all intervals I_i, which affects constraint (6.3). However, one can show that the constraint matrix of this LP is generally not totally unimodular. We illustrate the behaviour of the LP with the help of the following exemplary instance in Figure 6.1, in which all edges have unit-size jobs associated and the label of an edge e represents (r_e, d_e). It is easy to verify that a schedule that preempts jobs only at integral time points, has maximum connectivity time of one. However, the following schedule with arbitrary preemption has connectivity time of two. We process $\{s^+, v_2\}$ in $[0, 0.5] \cup [1, 1.5]$, $\{s^+, v_3\}$ in $[0.5, 1] \cup [1.5, 2]$, $\{v_4, s^-\}$ in $[0, 0.5] \cup [1.5, 2]$, $\{v_5, s^-\}$ in $[0.5, 1.5]$, and the other edges are fixed by their time window. This instance shows that the integrality gap of the LP is at least two.

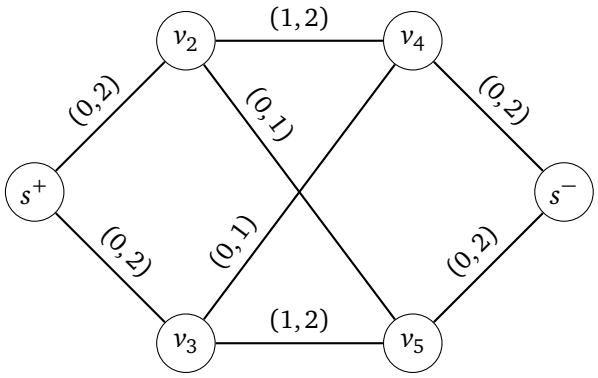


Figure 6.1: Example for the difference between arbitrary preemption and preemption only at integral time points.

The statement of Theorem 6.1 crucially relies on the fact that we may preempt jobs arbitrarily. However, if preemption is only possible at integral time points, the problem becomes $\mathcal{NP}$-hard even for unit-size jobs. This follows from the proof of Theorem 6.5 for $t_1 = 0$, $t_2 = 1$, and $T = 2$.

Theorem 6.4 MAXCONNECTIVITY *with preemption only at integral time points is NP-hard and does not admit a* $(2-\epsilon)$*-approximation algorithm for any* $\epsilon > 0$*, unless* $\mathcal{P} = \mathcal{NP}$*. Furthermore,* MINCONNECTIVITY *with preemption only at integral time points is inapproximable.*

6.3 Non-Preemptive Scheduling

We consider problem instances in which no job can be preempted. We show that there is no $(c\sqrt[3]{|E|})$-approximation algorithm for MAXCONNECTIVITY for some $c > 0$. We also show that MINCONNECTIVITY is inapproximable, unless $\mathcal{P} = \mathcal{NP}$. Furthermore, we give an $(\ell + 1)$-approximation algorithm, where $\ell := |\{d_e - p_e \mid e \in E\}|$ is the number of distinct latest start times for jobs.

To show the strong hardness of approximation for MAXCONNECTIVITY, we begin with a weaker result which provides us with a crucial gadget.

Theorem 6.5 *Non-preemptive* MAXCONNECTIVITY *does not admit a* $(2-\epsilon)$*-approximation algorithm, for* $\epsilon > 0$*, and non-preemptive* MINCONNECTIVITY *is inapproximable, unless* $\mathcal{P} = \mathcal{NP}$*. This holds even for unit-size jobs.*

Proof. We show that the existence of a $(2-\epsilon)$-approximation algorithm for non-preemptive MAXCONNECTIVITY allows to distinguish between YES- and NO-instances of 3SAT in polynomial time. Given an instance of 3SAT consisting of m clauses $C_1, C_2, \ldots, C_m$ each of exactly three variables in $X = \{x_1, x_2, \ldots, x_n\}$, we construct the following instance of non-preemptive MAXCONNECTIVITY. We pick two arbitrary but distinct time points $t_1 + 1 \leq t_2$ and a polynomially bounded time horizon $T \geq t_2 + 1$. We construct our instance such that connectivity is impossible outside $[t_1, t_1 + 1]$ and $[t_2, t_2 + 1]$. For this, s^+ is followed by a path P from s^+ to a vertex s' composed of three edges that disconnect s^+ from s^- in the time intervals $[0, t_1]$, $[t_1 + 1, t_2]$, and $[t_2 + 1, T]$. These edges e have $p_e = d_e - r_e$. Furthermore, we construct the network such that the total connectivity time is greater than one if and only if the 3SAT-instance is a YES-instance. And we show that if the total connectivity time is greater than one, then there is a schedule with maximum total connectivity time of two. The high-level structure of the graph we will be creating can be found in Figure 6.2, with an expanded version following later in Figure 6.3.

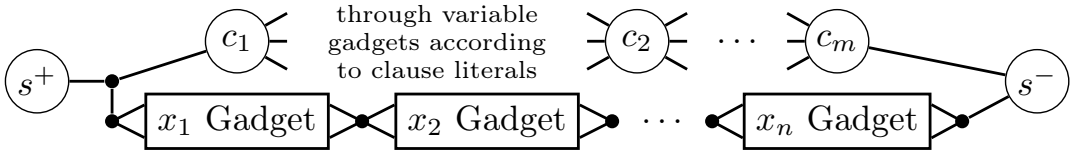


Figure 6.2: High-level view of the construction for Theorem 6.5.

Let $Y(x_i)$ be the set of clauses containing the literal x_i and $Z(x_i)$ be the set of clauses containing the literal $\neg x_i$, and set set $k_i = 2|Y(x_i)|$ and $\ell_i = 2|Z(x_i)|$. We define the following node sets

- $V_1 := \{y_i^1, \ldots, y_i^{k_i} \mid i = 1, \ldots, n\}$,
- $V_2 := \{z_i^1, \ldots, z_i^{\ell_i} \mid i = 1, \ldots, n\}$,
- $V_3 := \{c_r \mid r = 1, \ldots, m+1\}$,
- $V_4 := \{v_i \mid i = 1, \ldots, n+1\}$
- and set $V = \bigcup_{j=1}^{4} V_j \cup \{v : v \in P\} \cup \{s^-\}$.

We introduce three edge types

- $\mathcal{E}_1 := \{e \in E : r_e = t_1, d_e = t_2 + 1, p_e = t_2 - t_1\}$,
- $\mathcal{E}_2 := \{e \in E : r_e = t_1, d_e = t_1 + 1, p_e = 1\}$,
- and $\mathcal{E}_3 := \{e \in E : r_e = t_2, d_e = t_2 + 1, p_e = 1\}$.

The graph $G = (V, E)$ consists of variable gadgets, shown in Figure 6.3, to which we connect the clause nodes c_r, $r = 1, \ldots, m+1$. We define the following edge sets for the variable gadgets, namely,

- $E_1 := \{\{s', v_1\}, \{v_{n+1}, s^-\}\}$ of type $\mathcal{E}_2$,
- $E_2 := \{\{v_i, y_i^1\}, \{v_i, z_i^1\}, \{y_i^{k_i}, v_{i+1}\}, \{z_i^{\ell_i}, v_{i+1}\} : i = 1, \ldots, n\}$ of type $\mathcal{E}_2$,
- $E_3 := \{\{y_i^q, y_i^{q+1}\} : i = 1, \ldots, n; q = 1, 3, \ldots, k_i - 3, k_i - 1\}$ of type $\mathcal{E}_1$,
- $E_4 := \{\{z_i^q, z_i^{q+1}\} : i = 1, \ldots, n; q = 1, 3, \ldots, \ell_i - 3, \ell_i - 1\}$ of type $\mathcal{E}_1$,
- $E_5 := \{\{y_i^q, y_i^{q+1}\} : i = 1, \ldots, n; q = 2, 4, \ldots, k_i - 4, k_i - 2\}$ of type $\mathcal{E}_2$,
- and $E_6 := \{\{z_i^q, z_i^{q+1}\} : i = 1, \ldots, n; q = 2, 4, \ldots, \ell_i - 4, \ell_i - 2\}$ of type $\mathcal{E}_2$.

Notice that a variable x_i may only appear positive ($\ell_i = 0$) or only negative ($k_i = 0$) in our set of clauses. In this case, we also have an edge of type $\mathcal{E}_2$ connecting v_i and v_{i+1} besides the construction for the negative (z nodes) or positive part (y nodes). Finally, we add edges to connect the clause nodes to the graph. If some positive literal x_i appears in clause C_r and C_r is the q-th clause with positive x_i, we add the edges $\{c_r, y_i^{2q-1}\}$ and $\{y_i^{2q}, c_{r+1}\}$ both of type $\mathcal{E}_3$. Conversely, if some x_i appears negated in C_r and C_r is the q-th clause with $\neg x_i$, we add the edges $\{c_r, z_i^{2q-1}\}$ and $\{z_i^{2q}, c_{r+1}\}$ both of type $\mathcal{E}_3$. We also connect c_1 and c_{m+1} to the graph by adding $\{s', c_1\}$ and $\{c_{m+1}, s^-\}$ of type $\mathcal{E}_3$. We define E to be the union of all introduced edges. Observe that the network G has $O(n+m)$ nodes and edges.

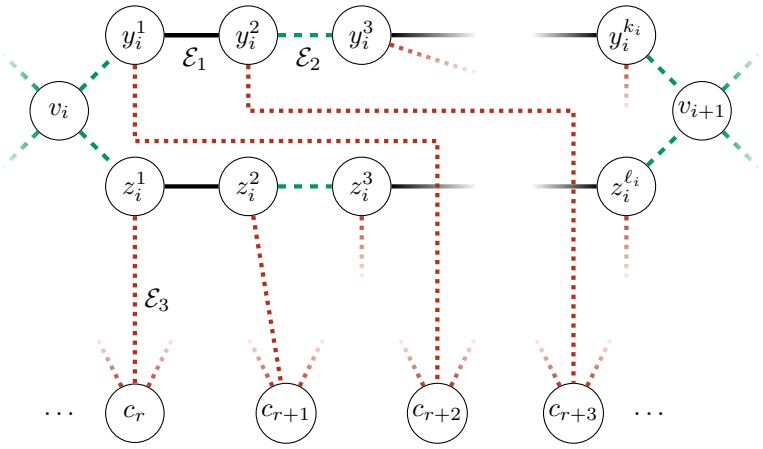


Figure 6.3: Schematic representation of the gadget for variable x_i, which appears negated in clause C_r and positive in clause C_{r+2} among others.

We call an (s^+, s^-)-path that contains no node from V_3 a *variable path* and an (s^+, s^-)-path with no node from V_4 a *clause path*. An (s^+, s^-)-path containing edges of type $\mathcal{E}_2$ and $\mathcal{E}_3$ does not connect s^+ with s^- in $[t_1, t_1 + 1]$ or in $[t_2, t_2 + 1]$. Therefore, all paths other than variable paths and relevant clause paths are irrelevant for the connectivity of s^+ with s^-.

When maintaining all edges of type $\mathcal{E}_1$ in $[t_1, t_2]$, we have connectivity in $[t_2, t_2 + 1]$ exactly on all variable paths. Conversely, maintaining all edges of type $\mathcal{E}_1$ in $[t_1 + 1, t_2 + 1]$ yields connectivity in $[t_1, t_1 + 1]$ exactly on all relevant clause paths. On the other hand, any clause path can connect s^+ with s^- only in $[t_1, t_1 + 1]$ and any variable path only in $[t_2, t_2 + 1]$. We now claim that there is a schedule with total connectivity time greater than one if and only if the 3SAT-instance is a YES-instance.

Let S be a schedule with total connectivity time greater than one. Then there is a variable path P^v with positive connectivity time in $[t_2, t_2+1]$ and a clause path P^c with positive connectivity time in $[t_1, t_1+1]$. As the total connectivity time is greater than one, P^c cannot walk through both the positive part (y nodes) and the negative part (z nodes) of the gadget for any variable x_i. This allows to assume w.l.o.g. that P^v and P^c are disjoint between s' and s^-. Say P^v and P^c share an edge on the negative part (z nodes) of the gadget for variable x_i. Then we can redirect the variable path P^v to the positive part (y nodes) without decreasing the total connectivity time. The same works if they share an edge on the positive part.

Now set x_i to FALSE if P^v uses the nodes $y_i^1, \dots, y_i^{k_i}$, that is the upper part of the variable gadget, and to TRUE otherwise. With this setting, whenever P^c uses edges of a variable gadget, e.g. the sequence $c_r, z_i^{2q-1}, z_i^{2q}, c_{r+1}$ for some r, q, disjointness of P^v and P^c implies that clause C_r is satisfied with the truth assignment of variable x_i. Since every node pair c_r, c_{r+1} is only connected with paths passing through variable gadgets, and at least one of them belongs to P^c we conclude that every clause C_r is satisfied.

Consider a satisfying truth assignment. We define a schedule that admits a variable path P^v with connectivity in $[t_2, t_2+1]$. This path P^v uses the upper part (y_i-part) if x_i is set to FALSE and the lower part (z_i-part) if x_i is set to TRUE. That is, we maintain all edges of type $\mathcal{E}_1$ on the upper path (y_i-path) of the variable gadget for x_i in $[t_1, t_2]$ if x_i is FALSE and in $[t_1+1, t_2+1]$ if x_i is TRUE. Conversely, edges of type $\mathcal{E}_1$ on the lower path (z_i-path) of the variable gadget for x_i are maintained in $[t_1, t_2]$ if x_i is TRUE and in $[t_1+1, t_2+1]$ if x_i is FALSE. This implies for the part of the gadget for x_i that is not used by P^v that the corresponding edges of type $\mathcal{E}_1$ are scheduled to allow connectivity during $[t_1, t_1+1]$. These edges can be used in a clause path to connect node c_r with c_{r+1} for some clauses C_r that is satisfied by the truth assignment of x_i. Since all clauses are satisfied by some variable x_i there exists a clause path P^c admitting connectivity in $[t_1, t_1+1]$. Therefore, the constructed schedule allows connectivity during both intervals $[t_1, t_1+1]$ and $[t_2, t_2+1]$.

To show the inapproximability of MINCONNECTIVITY, we reduce 3SAT to this problem. We construct an instance of MINCONNECTIVITY exactly the same way as we did above for MAXCONNECTIVITY and set $t_1 = 0$, $t_2 = 1$, and $T = 2$. By definition of the jobs, this results in a instance with only unit-sized jobs. As we discussed above, YES-instances of 3SAT result in a MAXCONNECTIVITY instance with an objective value of 2. For $T = 2$, that means we have connectivity at all time points, and therefore an objective value of 0 for MINCONNECTIVITY. NO-instances of 3SAT on the other result in MAXCONNECTIVITY instance with an objective value of 1 – for $T = 2$, this results in MINCONNECTIVITY objective value of 1 as well. Due to the gap between 1 and 0, any approximation algorithm that outputs a solution within a factor of the optimum solution needs to decide 3SAT. □

We reuse the construction in the proof of Theorem 6.5 repeatedly to obtain the following improved lower bound.

Theorem 6.6 *Unless $\mathcal{P} = \mathcal{NP}$, there is no $(c\sqrt[3]{|E|})$-approximation algorithm for non-preemptive* MAXCONNECTIVITY*, for some constant $c > 0$.*

Proof. We reuse the construction in the proof of Theorem 6.5 to construct a network that has maximum connectivity time n if the given 3SAT instance is a YES-instance and maximum connectivity time 1 otherwise. This implies that there cannot be an $(n-\epsilon)$-approximation

algorithm for non-preemptive MAXCONNECTIVITY, unless $\mathcal{P} = \mathcal{NP}$. Here, n is again the number of variables in the given 3SAT instance. Note that the construction in the proof of Theorem 6.5 has $\Theta(n)$ maintenance jobs and thus there exists a constant $c_1 > 0$ such that $|E| \leq c_1 \cdot n$. In this proof, we will introduce $\Theta(n^2)$ copies of the construction and thus $|E| \leq c_2 \cdot n^3$ for some $c_2 > 0$, which gives that $n \geq c_3 \sqrt[3]{|E|}$ for some $c_3 > 0$. This gives the statement.

For the construction, we use $n^2 - n$ copies of the 3SAT-network from the proof of Theorem 6.5, where each one uses different (t_1, t_2)-combinations with $t_1, t_2 \in \{0, \ldots, n-1\}$ and $t_1 \neq t_2$. We use these copies as 3SAT-gates and mutually connect them as depicted in Figure 6.4. Recall that for one such 3SAT-network we have the freedom of choosing the intervals $[t_1, t_1 + 1]$ and $[t_2, t_2 + 1]$, which are relevant for connectivity. This choice now differs for every 3SAT-gate.

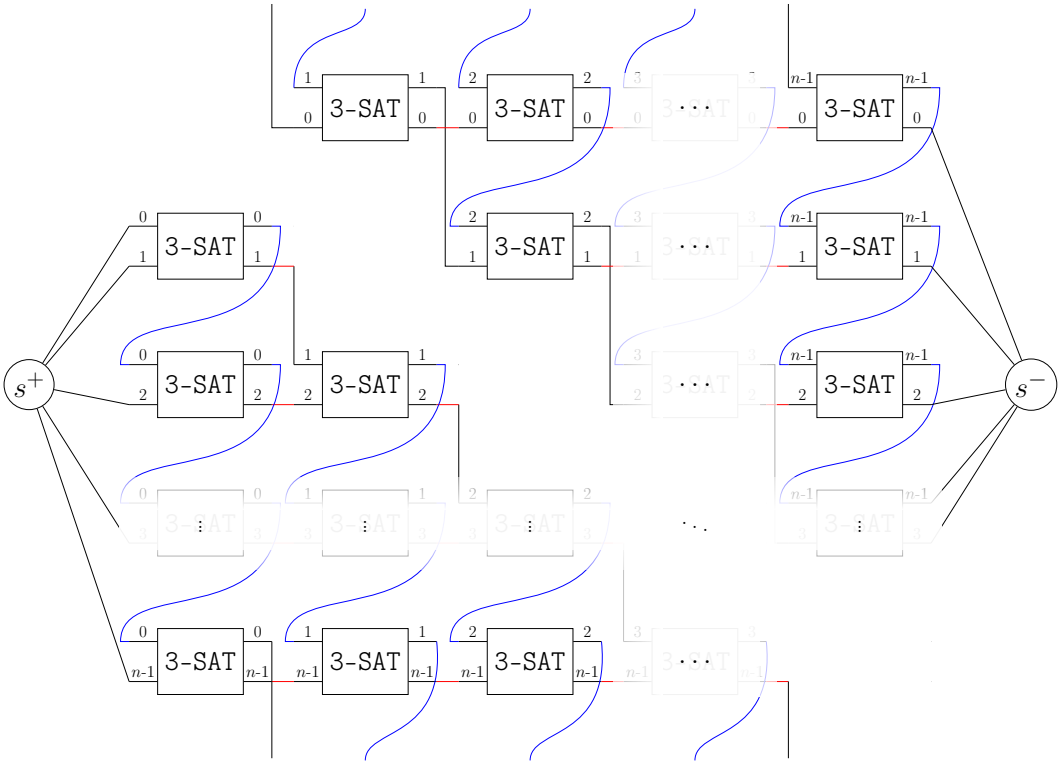


Figure 6.4: Schematic representation of the network of 3SAT-gates.

Think of the construction as an $(n \times n)$-matrix M with an empty diagonal. Entry (i, j), $i, j \in \{0, \ldots, n-1\}$, in M corresponds to a 3SAT-gate in that variable paths only exist in time slot $[i, i+1]$ and relevant clause paths exist only in $[j, j+1]$. This is enforced by the edges of type $\mathcal{E}_2$, which prevent variable paths in $[j, j+1]$, and edges of type $\mathcal{E}_3$, which prevent relevant clause paths in $[i, i+1]$. Edges between the s^+-copy and s'-copy of the 3SAT-gate(i, j) prevent connectivity outside of $[i, i+1]$ and $[j, j+1]$. Note that now $\mathcal{E}_1 := \{e \in E : r_e = i, d_e = j+1, p_e = j-i\}$ if $i < j$, and $\mathcal{E}_1 := \{e \in E : r_e = j, d_e = i+1, p_e = i-j\}$ if $i > j$.

The s^+-copy of the 3SAT-gate(i, j) is connected to two paths, where one of them allows connectivity only during $[i, i+1]$ and the other one only during $[j, j+1]$. The same is done

for the s^--copy of the 3SAT-gate(i,j). In Figure 6.4, this is illustrated by labels on the paths. A label $i \in \{0,\dots,n-1\}$ means, that this path allows connectivity only during $[i,i+1]$. The upper path connected to a 3SAT-gate specifies the time slot, where variable paths may exist, and the lower path specifies the time slot, where relevant clause paths may exist. When following the path with label $k \in \{0,\dots,n-1\}$, we pass the gadgets in column $j = 0,\dots,k-1$ on the lower path having j on the upper path. In column k, we walk through all gadgets on the upper path and then we proceed with column $j = k+1,\dots,n-1$ on the lower path having j again on the upper path. Eventually, we connect the 3SAT-gate$(n-1,k)$ to the vertex s^-.

Note that within 3SAT-gate(i,j) we have connectivity during $[i,i+1]$ *and* $[j,j+1]$ if and only if the corresponding 3SAT-instance is a YES-instance. Also notice that we can assume due to [BKK15b] that all jobs start at integral times, which allows us to ignore schedules with fractional job starting times and therefore fractional connectivity within a time interval $[i,i+1]$. Now, if the 3SAT-instance is a YES-instance, there is a global schedule such that its restriction to every 3SAT-gate(i,j) allows connectivity during both intervals. Thus for each label $k \in \{0,\dots,n-1\}$ there exists a path with this label that has connectivity during $[k,k+1]$. This implies that the maximum connectivity time is n.

Conversely, suppose there exists a global schedule with connectivity during $[i,i+1]$ and $[j,j+1]$ for some $i \neq j$. Then there must exist two paths P_1, P_2 from s^+ to s^- with two distinct labels i and j, each realizing connectivity during one of both intervals. By construction they walk through the 3SAT-gate(i,j). This implies by the proof of Theorem 6.5, that the global schedule restricted to this gate corresponds to a satisfying truth assignment for the 3SAT-instance. That is, the 3SAT-instance is a YES-instance. With the previous observation, it follows that an optimal schedule has maximum connectivity time of n. □

The results above hold for general graph classes, but even for graphs as simple as disjoint paths between s and t, the problem remains strongly $\mathcal{NP}$-hard.

Theorem 6.7 *Non-preemptive* MAXCONNECTIVITY *is strongly $\mathcal{NP}$-hard, and non-preemptive* MINCONNECTIVITY *is inapproximable even if the given graph consists only of disjoint paths between s and t.*

Proof. We proof this result by reduction from the strongly $\mathcal{NP}$-complete 3SAT problem.

> 3SAT
> **Input:** Clauses $C_1,\dots,C_m$ of exactly three variables in $x_1,\dots,x_n$.
> **Problem:** Is there a truth assignment to the variables in $x_1,\dots,x_n$ that satisfies all clauses?

We construct a network with $2n$ paths from s^+ to s^-, two for each variable of the 3SAT instance. Let P_i and $\bar{P}_i$ denote the two paths for variable x_i. We will introduce several maintenance jobs for each path, understanding that each new job is associated with a different edge of the path. Since the ordering of these edges does not matter, we will directly associate each job with a path without explicitly specifying the respective edge of the job. The network will allow a schedule that maintains connectivity at all times if and only if the 3SAT instance is satisfiable.

For convenience, assume that $n \geq m$, otherwise we introduce additional dummy variables. We define a time horizon $T = 8n$ that we subdivide into five intervals $A = [0, 2n), B = [2n, 3n), C = [3n, 5n), D = [5n, 6n), E = [6n, 8n]$. We will use these intervals now when defining jobs.

Jobs representing variables. For each variable x_i, we define a job each on paths P_i and $\bar{P}_i$ with the time window $[0, T]$ and processing time $3n$. We will ensure that neither job is scheduled to cover the time interval C entirely in any feasible schedule for the connectivity problem. This implies that a variable job either covers B or D without intersecting the other. The job on P_i (resp. $\bar{P}_i$) covering B will correspond to the literal x_i (resp. $\bar{x}_i$) being set to TRUE. We will of course ensure that not both literals can be set to TRUE simultaneously, but we will allow both to be FALSE, which simply means that the truth assignment remains satisfying, no matter how the variable is set.

Jobs needed to translate schedules into variable assignments. In the following, we introduce blocking jobs that all have a time window of unit length and unit processing time. In this way, introducing a blocking job at time t simply renders the corresponding path unusable during the time interval $[t, t+1)$. To ensure that the variable jobs for variable x_i do not cover C completely, we add a blocking job at time $t_i = 3n + 2(i-1)$ to all paths except P_i and a blocking job at time $t_i' = 3n + 2(i-1) + 1$ to all paths except $\bar{P}_i$. The first job forces the variable job for the literal x_i not to cover C completely, since otherwise connectedness is interrupted during the time interval $[t_i, t_i')$. The second blocking job accomplishes the same for the literal $\bar{x}_i$. Note that the blocking jobs for each literal occupy a unique part of the time window C.

Jobs preventing variables from being 0 and 1 at the same time. In order to force at most one literal of each variable x_i to be set to TRUE, we introduce a blocking job at time $t_i'' = 2n + (i-1)$ on all paths except P_i and $\bar{P}_i$. These blocking jobs ensure that either path P_i or $\bar{P}_i$ must be free during time $[t_i'', t_i'' + 1)$, which means not both variable jobs may be scheduled to cover B (recall each variable job either covers B or D without intersecting the other). Again, the blocking jobs for each variable occupy a unique part of the time window B.

Jobs enforcing that at least one literal of each clause is true. For each clause C_j we introduce a blocking job at time $5n + j$ on each path except the three paths that correspond to literals in C_j. Figure 6.5 shows this construction for variable x_i and paths $P_i, \bar{P}_i$.

These blocking jobs force that at least one of the literals of the clause has to be set to TRUE, i.e., be scheduled to overlap B instead of D, otherwise connectivity is interrupted during time $[5n + j, 5n + j + 1)$. Note again that the blocking jobs for each clause occupy a unique part of the time window D.

It is now easy to verify that each satisfying truth assignment leads to a feasible schedule without disconnectedness for the connectivity problem and vice versa.

We can use this instance construction for both MAXCONNECTIVITY and MINCONNECTIVITY. On the one hand, we have that YES-instances of 3SAT result in instances with a MAXCONNECTIVITY objective value of T and a MINCONNECTIVITY objective value of 0, and on the other hand we have that NO-instances of 3SAT result in instances with a

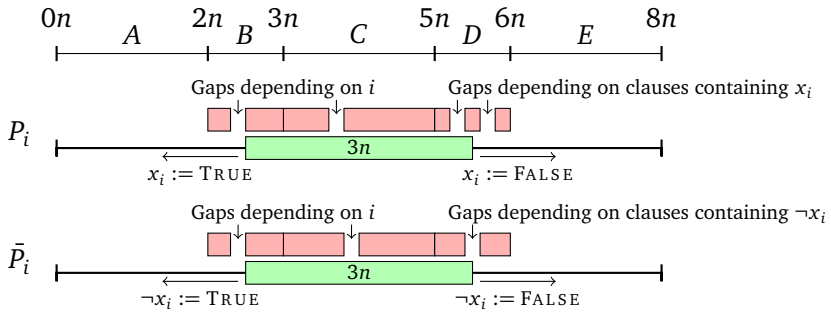


Figure 6.5: The paths $P_i, \bar{P}_i$ for variable x_i. The axis marks the times from 0 to $8n$.

MAXCONNECTIVITY objective value $< T$ and a MINCONNECTIVITY objective value > 0. This gives us the strong $\mathcal{NP}$-hardness for MAXCONNECTIVITY; for MINCONNECTIVITY, we get the inapproximability result since the optimal objective value is 0 here, similar to Theorem 6.5. □

We give an algorithm that computes an $(\ell+1)$-approximation for non-preemptive MAXCONNECTIVITY, where $\ell \leq |E|$ is the number of different time points $d_e - p_e, e \in E$. The basic idea is that we consider a set of $\ell+1$ feasible maintenance schedules, whose total time of connectivity upper bounds the maximum total connectivity time of a single schedule. Then the schedule with maximum connectivity time among our set of $\ell+1$ schedules is an $(\ell+1)$-approximation.

The schedules we consider start every job either immediately at its release date, or at the latest possible time. In the latter case it finishes exactly at the deadline. More precisely, for a fixed time point t, we start the maintenance of all edges $e \in E$ with $d_e - p_e \geq t$ at their latest possible start time $d_e - p_e$. All other edges start maintenance at their release date r_e. This yields at most $\ell + 1 \leq |E| + 1$ different schedules S_t, as for increasing t, each time point where $d_e - p_e$ is passed for some edge e defines a new schedule. Algorithm 6.1 formally describes this procedure, where $E(t) := \{e \in E : e \text{ is not maintained at } t\}$.

Algorithm 6.1 considers finitely many intervals, as all (sub-)interval bounds are defined by a time point $r_e, r_e + p_e, d_e - p_e$ or d_e of some $e \in E$. As we can check the network for (s^+, s^-)-connectivity in polynomial time, and the algorithm does this for each (sub-)interval, Algorithm 6.1 runs in polynomial time.

Theorem 6.8 *Algorithm 6.1 is an $(\ell+1)$-approximation algorithm for non-preemptive* MAXCONNECTIVITY *on general graphs, with $\ell \leq |E|$ being the number of different time points $d_e - p_e, e \in E$.*

Proof. By construction, all schedules $S_i, i = 1, \ldots, \ell+1$, are feasible and the solution returned has a connectivity time of $\max_{i=1,\ldots,\ell+1} c(t_i)$, with $c(t_i)$ being the connectivity time of schedule S_i.

The schedule S_i, $i = 1, \ldots, \ell+1$ is choosen in such a way that the connected time in the interval $[t_{i-1}, t_i]$ is maximized. To see this, we need to consider two types of jobs. First, all jobs on edges $e \in E$ with $d_e - p_e \geq t_i$ can be scheduled outside of $[t_{i-1}, t_i]$, which is definitely a correct choice in order to maximize the connectivity time in $[t_{i-1}, t_i]$. Second, for

Algorithm 6.1: Approx. Algorithm for Non-preemptive MAXCONNECTIVITY

Let $t_1 < \cdots < t_\ell$ be all different time points $d_e - p_e, e \in E$, $t_0 = 0$ and $t_{\ell+1} = T$.
Let S_i be the schedule, where all edges e with $d_e - p_e < t_i$ start maintenance at r_e and all other edges at $d_e - p_e$, $i = 1, \ldots, \ell + 1$.
For each S_i, initialize total connectivity time $c(t_i) \leftarrow 0, i = 1, \ldots, \ell + 1$.
for $i = 1$ to $\ell + 1$ **do**
    Partition the interval $[t_{i-1}, t_i]$ into subintervals such that each time point $r_e, r_e + p_e, d_e$, $e \in E$, in this interval defines a subinterval bound.
    for all subintervals $[a, b] \subseteq [t_{i-1}, t_i]$ **do**
        if $(V, E(1/2 \cdot (a + b)))$ contains an (s^+, s^-)-path for S_i **then**
            Increase $c(t_i)$ by $b - a$.
        end if
    end for
end for
return Schedule S_i for which $c(t_i), i = 1, \ldots, \ell + 1$, is maximized.

all edges $e \in E$ with $d_e - p_e < t_i$, we know due to the definition of t_{i-1} that $r_e \leq d_e - p_e \leq t_{i-1}$. Thus, scheduling these jobs at r_e guarantees the least reduction in connectivity time in $[t_{i-1}, t_i]$. More precisely, this scheduling disrupts connectivity in the interval $[t_{i-1}, r_e + p_e]$ if $t_{i-1} \leq r_e + p_e$, and otherwise not at all. However, all other feasible schedulings must also disrupt connectivity in this interval – scheduling the job earlier than r_e is not possible, and neither is scheduling the job later than $d_e - p_e \leq t_{i-1}$. Thus, schedule S_i has the maximal connectivity time in $[t_{i-1}, t_i]$.

Since the intervals $[t_{i-1}, t_i]$, $i = 1, \ldots, \ell + 1$ partition the complete time window $[0, T]$, this allows us to bound the value of the optimal solution OPT by

$$\text{OPT} \leq \sum_{i=1}^{\ell+1} c(t_i) \leq (\ell + 1) \max_{i=1,\ldots,\ell+1} c(t_i) = (\ell + 1)\text{ALG} \tag{6.8}$$

with ALG being the value of a solution returned by Algorithm 6.1. This gives us an approximation guarantee of $\ell + 1$ and completes our proof. □

6.4 Power of Preemption

We first focus on MINCONNECTIVITY on a path and analyze how much we can gain by allowing preemption. First, we show that there is an algorithm that computes a non-preemptive schedule whose value is bounded by $O(\log |E|)$ times the value of an optimal preemptive schedule. Second, we argue that one cannot gain more than a factor of $\Omega(\log |E|)$ by allowing preemption.

Theorem 6.9 *The power of preemption is* $\Theta(\log |E|)$ *for* MINCONNECTIVITY *on a path.*

Proof. Observe that if at least one edge of a path is maintained at time t, then the whole path is disconnected at t. We give an algorithm for MINCONNECTIVITY on a path that

constructs a non-preemptive schedule with cost at most $O(\log |E|)$ times the cost of an optimal preemptive schedule.

We first compute an optimal preemptive schedule. This can be done in polynomial time by Theorem 6.1. Let x_t be a variable that is 1 if there exists a job j that is processed at time t and 0 otherwise. We shall refer to x also as the *maintenance profile*. Furthermore, let $a := \int_0^T x_t \, \mathrm{d}t$ be the active time, i.e., the total time of maintenance. Then we apply the following *splitting procedure*. We compute the time point $\bar{t}$ where half of the maintenance is done, i.e., $\int_0^{\bar{t}} x_t \, \mathrm{d}t = a/2$. Let $E(t) := \{e \in E \mid r_e \leq t \wedge d_e \geq t\}$ and $p_{\max} := \max_{e \in E(t)} p_e$. We reserve the interval $[\bar{t} - p_{\max}, \bar{t} + p_{\max}]$ for the maintenance of the jobs in $E(\bar{t})$, although we might not need the whole interval. We schedule each job in $E(\bar{t})$ around $\bar{t}$ so that the processing time before and after $\bar{t}$ is the same. If the release date (deadline) of a jobs does not allow this, then we start (complete) the job at its release date (deadline). Then we mark the jobs in $E(\bar{t})$ as scheduled and delete them from the preemptive schedule.

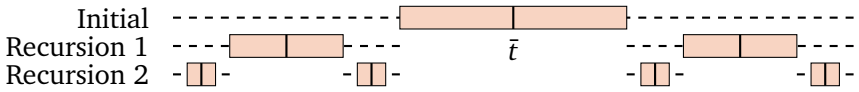


Figure 6.6: A sketch of the splitting procedure and the reserved intervals.

This splitting procedure splits the whole problem into two separate instances $E_1 := \{e \in E \mid d_e < \bar{t}\}$ and $E_2 := \{e \in E \mid r_e > \bar{t}\}$. Note that in each of these sub-instances the total active time in the preemptive schedule is at most $a/2$. We apply the splitting procedure to both sub-instances and follow the recursive structure of the splitting procedure until all jobs are scheduled. □

Lemma 6.10 *For* MinConnectivity *on a path, the given algorithm constructs a non-preemptive schedule with cost $O(\log |E|)$ times the cost of an optimal preemptive schedule.*

Proof. The progression of the algorithm can be described by a binary tree in which a node corresponds to a partial schedule generated by the splitting procedure for a subset of the job and edge set E. The root node corresponds to the partial schedule for $E(\bar{t})$ and the (possibly) two children of the root correspond to the partial schedules generated by the splitting procedure for the two subproblems with initial job sets E_1 and E_2. We can cut a branch if the initial set of jobs is empty in the corresponding subproblem. We associate with every node v of this tree B two values (s_v, a_v) where s_v is the number of scheduled jobs in the subproblem corresponding to v and a_v is the amount of maintenance time spent for the scheduled jobs.

The binary tree B has the following properties. First, $s_v \geq 1$ holds for all $v \in B$, because the preemptive schedule processes some job at the midpoint $\bar{t}_v$ which means that there must be a job $e \in E$ with $r_e \leq \bar{t}_v \wedge d_e \geq \bar{t}_v$. This observation implies that the tree B can have at most $|E|$ nodes and since we want to bound the worst total cost we can assume w.l.o.g. that B has exactly $|E|$ nodes. Second, $\sum_{v \in B} a_v = \int_0^T y_t \, \mathrm{d}t$ where y_t is the maintenance profile of the non-preemptive solution.

The cost a_v of the root node (level-0 node) is bounded by $2p_{\max} \leq 2a$. The cost of each level-1 node is bounded by $2 \cdot a/2 = a$, so the total cost on level 1 is also at most $2a$. It is easy to verify that this is invariant, i.e., the total cost at level i is at most $2a$ for all $i \geq 0$, since the

worst node cost a_v halves from level i to level $i+1$, but the number of nodes doubles in the worst case. We obtain the worst total cost when B is a complete balanced binary tree. This tree has at most $O(\log|E|)$ levels and therefore the worst total cost is $a \cdot O(\log|E|)$. The total cost of the preemptive schedule is a. □

We now provide a matching lower bound for the power of preemption on a path.

Lemma 6.11 *The power of non-preemption is* $\Omega(\log|E|)$ *for* MINCONNECTIVITY *on a path.*

Proof. We construct a path with $|E|$ edges and divide the $|E|$ jobs into ℓ levels such that level i contains exactly i jobs for $1 \leq i \leq \ell$. Hence, we have $|E| = \ell(\ell+1)/2$ jobs. Let P be a sufficiently large integer such that all of the following numbers are integers. Let the jth job of level i have release date $(j-1)P/i$, deadline $(j/i)P$, and processing time P/i, where $1 \leq j \leq i$. Note that now no job has flexibility within its time window, and thus the value of the resulting schedule is P.

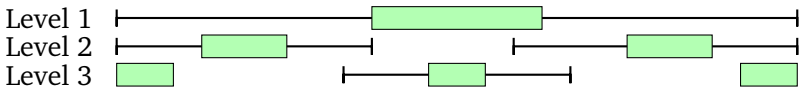


Figure 6.7: A rough sketch of the instance for 3 levels.

We now modify the instance as follows. At every time point t where at least one job has a release date and another job has a deadline, we stretch the time horizon by inserting a gap of size P. This stretching at time t can be done by adding a value of P to all time points after the time point t, and also adding a value of P to all release dates at time t. The deadlines up to time t remain the same. Observe that the value of the optimal preemptive schedule is still P, because when introducing the gaps we can move the initial schedule accordingly such that we do not maintain any job within the gaps of size P. Figure 6.7 shows a rough sketch of this construction.

We now consider the optimal non-preemptive schedule. The cost of scheduling the only job at level 1 is P. In parallel to this job we can schedule at most one job from each other level, without having additional cost. This is guaranteed by the introduced gaps. At level 2 we can fix the remaining job with additional cost $P/2$. As before, in parallel to this fixed job, we can schedule at most one job from each level i where $3 \leq i \leq \ell$. Applying the same argument to the next levels, we notice that for each level i we introduce an additional cost of value P/i. Thus the total cost is at least $\sum_{i=1}^{\ell} P/i \in \Omega(P \log \ell)$ with $\ell \in \Theta(\sqrt{|E|})$. □

Next, we show that for MAXCONNECTIVITY, the power of preemption can be unbounded.

Theorem 6.12 *For non-preemptive* MAXCONNECTIVITY *on a path the power of preemption is unbounded.*

Proof. Consider a path of four consecutive edges $e_1 = \{s^+, u\}, e_2 = \{u, w\}, e_3 = \{w, v\}, e_4 = \{v, s^-\}$, each associated with a maintenance job as depicted in Figure 6.8. That is, $r_1 = r_2 = 0, d_1 = r_3 = p_1 = p_4 = 1, p_2 = p_3 = 2, r_4 = d_2 = 3, d_3 = d_4 = 4$.

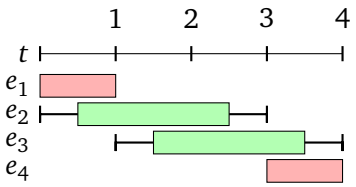


Figure 6.8: Example for an unbounded power of preemption.

There is no non-preemptive schedule that allows connectivity at any point in time, as the maintenance job of edge e_i blocks edge e_i in time slot $[i-1,i]$. On the other hand, when allowing preemptive schedules, we can process the job of edge e_2 in $[0,2]$ and the job of edge e_3 in $[1,2]$ and $[3,4]$. Then no maintenance job is scheduled in the time interval $[2,3]$ and therefore we have connectivity for one unit of time. □

6.5 Mixed Scheduling

We know that both the non-preemptive and preemptive MAXCONNECTIVITY and MINCONNECTIVITY on a path are solvable in polynomial time by Theorem 6.1 and [Kha+15, Theorem 9], respectively. Notice that the parameter g in [Kha+15] is in our setting ∞. Interestingly, the complexity changes when mixing the two job types – even on a simple path.

Theorem 6.13 MAXCONNECTIVITY *and* MINCONNECTIVITY *with preemptive and non-preemptive maintenance jobs is weakly $\mathcal{NP}$-hard, even on a path.*

Proof. We reduce the $\mathcal{NP}$-hard PARTITION problem to MAXCONNECTIVITY. We will show that there is a gap in the objective value between instances derived from YES- and NO-instances of PARTITION, respectively. This gap is same for MINCONNECTIVITY, since maximizing the time in which we have connectivity is the same as minimizing the time in which we do not have connectivity.

PARTITION
Input: A set of n natural numbers $A = \{a_1, \dots, a_n\} \subset \mathbb{N}$ with $\sum_{i=1}^{n} a_i = 2B$ for some $B \in \mathbb{N}$.
Problem: Is there a subset $S \subseteq A$ with $\sum_{a \in S} a = B$?

Given an instance of PARTITION, we create a MAXCONNECTIVITY instance based on a path consisting of $3n+2$ edges between s^+ and s^- with preemptive and non-preemptive maintenance jobs. We create three types of job sets denoted as J_1, J_2 and J_3, where the first two job sets model the binary decision involved in choosing a subset of numbers to form a partition, whereas the third job set performs the summation over the numbers picked for a partition. The high-level idea is depicted in Figure 6.9.

The job set $J_1 := \{1, 2, \dots, 2n-1, 2n\}$ contains $2n$ *tight* jobs, i.e., $r_j + p_j = d_j$ for all $j \in J_1$. For every element $a_i \in A$ we have two tight jobs i and $2n-(i-1)$ both having processing time $4^{n-i}B =: x_i$. The release date of a job $j \in \{2, \dots, n\} \subset J_1$ is $r_j = \sum_{k=1}^{j-1} 2x_k + a_k$ and $r_1 = 0$. Let $\tau := \sum_{k=1}^{n} 2x_k + a_k$. For $j \in \{n+1, \dots, 2n\} \subset J_1$ we have $d_j = \tau + \sum_{k=n+1}^{j} 2x_{2n-k+1} + a_{2n-k+1}$.

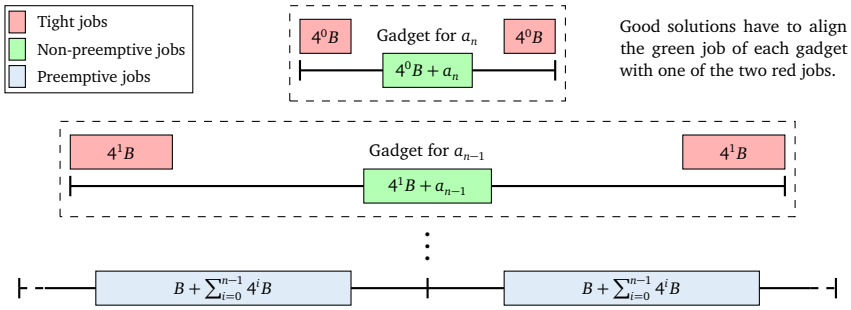


Figure 6.9: Instance created from a PARTITION instance $a_1, \ldots, a_n, B$. The number inside the blocks are the processing times of the jobs.

Note that the tight jobs in J_1 are constructed in such a way that everything is symmetric with respect to the time point τ.

The job set $J_2 := \{2n+1, \ldots, 3n\}$ contains n non-preemptive jobs. Let $j_i := 2n+i$. For every element $a_i \in A$ we introduce job j_i with processing time $p_{j_i} = x_i + a_i$, release date $r_{j_i} = r_i$, and deadline $d_{j_i} = d_{2n-(i-1)}$. Again, everything is symmetric with respect to time point τ.

Finally, the set $J_3 := \{3n+1, 3n+2\}$ contains two preemptive jobs, where each of them has processing time $W := B + \sum_{i=1}^{n} x_i$. Furthermore, we have $r_{3n+1} = 0$, $d_{3n+1} = \tau$, $r_{3n+2} = \tau$, $d_{3n+2} = 2\tau$.

We now show that there is a feasible schedule for the constructed instance that disconnects the path for at most $2W$ time units if and only if the given PARTITION instance is a YES-instance.

Suppose there is a subset $S \subseteq A$ with $\sum_{a \in S} a = B$. For each $a_i \in S$, we start the corresponding job $j_i \in J_2$ at its release date and the remaining jobs in J_2 corresponding to the elements $a_i \in A \setminus S$ are scheduled such that they complete at their deadline. This creates $B + \sum_{i=1}^{n} x_i$ time slots in both intervals $[0, \tau]$ and $[\tau, 2\tau]$ with no connection between s^+ and s^-. The jobs $3n+1$ and $3n+2$ can be preempted in $[0, \tau]$ and $[\tau, 2\tau]$, respectively, and thus if we align their processing with the chosen maintenance slots, we get a schedule that disconnects s^+ and s^- for $2W = 2(B + \sum_{i=1}^{n} x_i)$ time units.

Conversely, suppose that there is a feasible schedule for the constructed instance that disconnects the path for at most $2W$ time units. By induction on i, we show that every job $j_i = 2n+i$ either starts at its release date or it completes at its deadline in such a schedule.

Consider the base case of $i = 1$. We first observe that w.l.o.g. job j_1 either starts at its release date or completes at its deadline or is scheduled somewhere in $[x_1, 2\tau - x_1]$. Suppose it starts somewhere in $(0, x_1)$ or completes somewhere in $(\tau - x_1, \tau)$. Then we do not increase the total time where the path is disconnected if we push job j_1 completely to the left or completely to the right. If we schedule job j_1 in $[x_1, 2\tau - x_1]$, then the total time where the path is disconnected is at least $3x_1 + a_1 > 2x_1 + x_1$. We will now show that $x_1 \geq 2(B + \sum_{k=2}^{n} x_k)$ for $n \geq 2$, which shows that the path is then disconnected for more than $2W$ time units, and thus job j_1 cannot be processed in $[x_1, 2\tau - x_1]$. The inequality is true

for $n \geq 2$, since

$$\begin{aligned} 2B + 2\sum_{k=2}^{n} x_k &= 2B(1 + \sum_{k=2}^{n} 4^{n-k}) \\ &= 2B(1 + \sum_{k=0}^{n-2} 4^k) \\ &= 2B(1 + 1/3(4^{n-1} - 1)) \\ &\leq 4^{n-1}B = x_1. \end{aligned}$$

This finishes the proof for $i = 1$.

Suppose, the statement is true for $i = 1, \ldots, \ell - 1$ with $\ell \in \{2, \ldots, n-1\}$. As in the base case, we can show that job j_ℓ either starts at its release date or completes at its deadline or is scheduled somewhere in $[r_{j_\ell} + x_\ell, d_{j_\ell} - x_\ell]$. If job j_ℓ is processed in $[r_{j_\ell} + x_\ell, d_{j_\ell} - x_\ell]$, then the total time where the path is disconnected is at least

$$\sum_{k=1}^{\ell-1}(2x_k + a_k) + 3x_\ell + a_\ell > \sum_{k=1}^{\ell} 2x_k + x_\ell.$$

Again, we will show that $x_\ell \geq 2(B + \sum_{k=\ell+1}^{n} x_k)$ for $\ell \in \{2, \ldots, n-1\}$, which shows that the path is then disconnected for more than $2W$ time units, and thus job j_ℓ cannot be processed in $[r_{j_\ell} + x_\ell, d_{j_\ell} - x_\ell]$. The inequality is true for $\ell \in \{2, \ldots, n-1\}$, since

$$\begin{aligned} 2B + 2\sum_{k=\ell+1}^{n} x_k &= 2B(1 + \sum_{k=\ell+1}^{n} 4^{n-k}) \\ &= 2B(1 + \sum_{k=0}^{n-\ell-1} 4^k) \\ &= 2B(1 + 1/3(4^{n-\ell} - 1)) \\ &\leq 4^{n-\ell}B = x_\ell. \end{aligned}$$

For $i = n$, we again use the fact that j_n either starts at its release date or completes at its deadline or is scheduled somewhere in $[r_{j_n} + x_n, d_{j_n} - x_n]$. If the latter case is true, then the total time where the path is disconnected is at least

$$\begin{aligned} \sum_{k=1}^{n-1}(2x_k + a_k) + 3x_n + a_n &= \sum_{k=1}^{n}(2x_k + a_k) + x_n \\ &> 2(B + \sum_{k=1}^{n} x_k) = 2W. \end{aligned}$$

There is a feasible schedule for the constructed instance that disconnects the path for at most $2(B + \sum_{k=1}^{n} x_k)$ time units. This means that in both $[0, \tau]$ and $[\tau, 2\tau]$ the path is disconnected for exactly $B + \sum_{k=1}^{n} x_k$ time units. Consider the set $S := \{i : j_i \text{ starts at its release date}\}$. We conclude that

$$\sum_{k=1}^{n} x_k + \sum_{k \in S} a_k = \sum_{k=1}^{n} x_k + \sum_{k \notin S} a_k = \sum_{k=1}^{n} x_k + B. \tag{6.9}$$

□

For MINCONNECTIVITY, running the optimal preemptive and non-preemptive algorithms on the respective job sets individually gives a 2-approximation.

Theorem 6.14 *There is a 2-approximation algorithm for* MINCONNECTIVITY *on a path with preemptive and non-preemptive maintenance jobs.*

Proof. Consider an optimal schedule S^* for the mixed instance and let $|S^*|$ be the total time of disconnectivity in S^*. Furthermore, let S^*_{np} (resp. S^*_p) be the restriction of S^* to only non-preemptive (resp. preemptive) jobs. Note that the schedule S^*_{np} (resp. S^*_p) is feasible for the corresponding non-preemptive (resp. preemptive) instance. We separate the preemptive from the non-preemptive jobs and obtain two separate instances. Solving them individually in polynomial time and combining the resulting two solutions S_{np} and S_p to a schedule S gives the claimed result, because $|S| \leq |S_{np}| + |S_p| \leq |S^*_{np}| + |S^*_p| \leq 2|S^*|$. □

6.6 Conclusion

Combining network flows with scheduling aspects is a very recent field of research. While there are solutions using IP based methods and heuristics, exact and approximation algorithms have not been considered extensively. We provide strong hardness results for connectivity problems, which is inherent to all forms of maintenance scheduling, and give algorithms for tractable cases.

In particular, the absence of $c\sqrt[3]{|E|}$-approximation algorithms for some $c > 0$ for general graphs indicates that heuristics and IP-based methods [BKK15a; BKK15b; Bol+14] are a good way of approaching this problem. An interesting open question is whether the inapproximability results carry over to series-parallel graphs, as the network motivating [BKK15a; BKK15b; Bol+14] is series-parallel. Our results on the power of preemption as well as the efficient algorithm for preemptive instances show that allowing preemption is very desirable. Thus, it could be interesting to study models where preemption is allowed, but comes at a cost to make it more realistic.

On a path, our results have implications for minimizing busy time, as we want to minimize the number of times where some edge on the path is maintained. Here, an interesting open question is whether the 2-approximation for the mixed case can be improved, e.g. by finding a pseudo-polynomial algorithm, a better approximation ratio, or conversely, to show an inapproximability result for it.

Chapter 7

Conclusions

This thesis explores possibilities and limitations of new models for network design in the context of logistics networks. We consider various network design, planning and flow problems and focus on theoretical insights as well as algorithms applicable in practice. Many of our models emerged from research projects in close collaboration with logistics experts at 4flow AG [4fl17], a logistics consultancy company. We achieve a highly detailed modeling by including multidimensional properties of commodities, which enable to model consolidation effects and economies of scale. Although multi-attribute models have been considered in the literature before, they were, to the best of our knowledge, not applied to logistics networks of large scale.

Our first model for tactical transportation planning integrates realistic transportation tariffs, delivery patterns, and inventory costs. Several algorithmic techniques have been devised to address the challenges associated with the specific instance structure.

Our solution techniques range from purely combinatorial algorithms to mixed integer programming approaches computing strong lower bounds. A very successful approach is our local search procedure that simultaneously re-routes flow of multiple commodities, while using a generic tariff selection algorithm for evaluating edge costs. Equipping the local search with different types of initial solutions, such as multicommodity flow patterns derived from a strengthened LP relaxation, or from purely combinatorial path-based approaches, yields very good solutions. They are within a single digit percent of the optimum on average when testing on a broad set of real-world instances.

Our algorithms can be used both in connection with standard MIP solvers for optimal solution quality, or as purely combinatorial algorithms, yielding competitive solutions without usage of third-party software. Hence, the broad spectrum of our approaches offers a flexible trade-off between solution quality, operating cost and computation time.

Motivated by the model for tactical transportation planning we provide for a theoretical background and introduce the CONSOLIDATION STEINER SUBGRAPH problem on general graphs. It requires each commodity to be routed unsplittably and focuses on consolidation effects while ignoring economies of scale and relaxing capacity restrictions. Hardness for this general problem follows from a special case known as MULTICOST STEINER SUBGRAPH in the literature.

Thus, we consider various special cases and sharpen the border between polynomial time solvable and $\mathcal{NP}$-hard problems. Among the structural insights we find that the special case UNIQUE PROPERTY ONE-TARIFF-TYPE on undirected graphs allows optima to be

cycle free. However, the problem remains $\mathcal{NP}$-hard even when only two demands and two properties are considered.

For the most general case, we introduce the notion of forest costs that in general overestimate costs. Additionally, we show that they are even exact for instances with only two consolidation layers. We also propose a dynamic program that finds solutions with optimal forest costs, but has an exponential running time in the number of demands.

A major contribution of this thesis are solvable models for robust optimization under demand uncertainties. Our models go beyond the state of the art in that they combine robust optimization with detailed modeling of consolidation effects and economies of scale.

A first robust model focuses on strategic route planning for a logistics network from a customer's perspective. We give an exact formulation for the robust counterpart of the resulting routing problem and prove that the adversary problem is already $\mathcal{NP}$-hard. However, by a simplification of the cost function and the adversary model, we obtain a model that can be solved on real instances and preserves enough of the key features to produce competitive solutions. To evaluated the robust costs of the routings found in our computational study, we use the exact, $\mathcal{NP}$-hard robust model.

Results from an evaluation on a large real-world instance show that robust solutions are suboptimal for nominal demands, but may still perform well for the historical average. On the other hand, they improve the worst-case costs by single digit percentages. This is independent of the precise level of robustness chosen for the optimization.

The findings suggest that there are two kinds of historical good routings: those that are also robust and those that are not robust. Therefore, it seems worthwhile in logistic practice to pursue robust optimization in strategic planning.

In a next step, we extend the k-median hub location problem to logistics networks and present an algorithmic toolkit to obtain robust solutions.

In the literature, it has been observed that k-median MILP formulations may exhibit a strong LP relaxation, which motivates to focus on the LP relaxation of our model. We extend the relatively new technique of column-and-row generation to our formulation, which has sets of primary, secondary and tertiary variables. To the best of our knowledge, this structure has not yet been solved by column-and-row generation before. We prove an optimality criterion for our formulation that relies on network flow duality. However, for large scale instances, we have to confine ourselves to heuristics, resulting in an LP-based hub search for k-median hub location.

We address two concepts for robustness: Demand robustness and incremental hub chains. For incremental hub chains, we test a flexible decremental framework on very large instances in our test set. By comparing incremental solutions with k-median solutions, which are obtained with the LP-based hub search, on two medium sized instances, we observe a price of being incremental that is almost negligible.

The LP-based hub search can be extended to the simplified robust model to obtain cost robust solutions for medium sized instances. To the best of our knowledge, a robust optimization for hub location with model of high detail and instances of comparable size has not been conducted before. In a computational study we consider various uncertainty sets showing different tradeoffs between the price of robustness and cost savings for worst-case cost. With robust solutions, worst-case cost can be reduced by up to 10 % for some of the uncertainty sets.

For larger instances, even the combinatorial algorithms show very long running times

for the deterministic setting. Identifying the large number of demands as bottleneck, we present a generic aggregation concept and test two aggregation techniques. This allows us to compute incremental hub chains even for the largest instance, which currently exceeds the capabilities of standard logistics software.

Finally, we addressed the combination of network flows with scheduling aspects—we call it maintenance scheduling—a very recent field of research. While there are solutions using IP based methods and heuristics, exact and approximation algorithms have not been considered extensively. We provide strong hardness results for connectivity problems, which is inherent to all forms of maintenance scheduling, and give algorithms for tractable cases.

For *preemptive* maintenance jobs, we show that we can solve both problems, MAXCONNECTIVITY and MINCONNECTIVITY, efficiently in arbitrary networks. However, under the additional restriction that we can preempt jobs only at integral points in time, the problem becomes $\mathcal{NP}$-hard.

For *non-preemptive* instances, we establish that there is no $(c\sqrt[3]{|E|})$-approximation algorithm for MAXCONNECTIVITY for some constant $c > 0$ and that MINCONNECTIVITY is inapproximable even on disjoint paths between two nodes s and t, unless $\mathcal{P} = \mathcal{NP}$. On the positive side, we provide an $(\ell + 1)$-approximation algorithm for MAXCONNECTIVITY in general graphs, where ℓ is the number of distinct latest starting times (deadline minus processing time) for jobs.

For path graphs, we show that *mixed* instances, which have both preemptive and non-preemptive jobs, are weakly $\mathcal{NP}$-hard. This hardness result is of particular interest, as both purely non-preemptive and purely preemptive instances can be solved efficiently on a path. Furthermore, we give a simple 2-approximation algorithm for mixed instances of MINCONNECTIVITY.

We also investigate the power of preemption and show that it is $\Theta(\log|E|)$ for MINCONNECTIVITY on a path and unbounded for MAXCONNECTIVITY on a path. This is in contrast to other scheduling problems, where the power of preemption is constant.

Appendix A

Detailed Computational Results

A.1 Detailed Computational Results for Incremental Hub Chains

Table A.1: Detailed numbers for Figure 5.4 from Subsection 5.3.2: Speedup techniques for local search (LS): Combining Edge Heuristic (EH), Commodity Heuristic (CH), and Threshold Heuristic (TH). Note that the instance X_Handel in this prelimanary test refers to a smaller instance, where already 12 hubs are preselected. All other instances refer to those described in Subsection 5.4.1

solver	**H_Auto_1** cost/ time (s)	**H_Auto_2** cost/ time (s)	**H_Auto_3** cost/ time (s)	**H_Auto_4** cost/ time (s)	**H_Auto_5** cost/ time (s)	**X_Handel** cost/ time (s)
LS-CH-TH	313 993	1 626 347	1 911 715	217 260	1 397 912	3 051 528
	56	820	1836	106	2369	6806
LS-TH	313 920	1 625 942	1 911 432	216 901	1 397 750	3 051 504
	52	819	1751	110	2247	5901
LS	313 217	1 567 906	1 900 238	214 176	1 371 726	2 969 747
	67	1521	4863	184	4800	24 656
LS-CH	314 166	1 570 686	1 903 336	215 090	1 374 190	2 987 671
	59	1361	4952	160	4778	19 734
LS-EH-CH	313 949	1 570 768	1 903 555	215 136	1 374 251	2 987 160
	59	1452	3973	148	4814	17 571
LS-EH-CH-TH	315 062	1 627 822	1 911 811	217 408	1 398 652	3 051 949
	48	755	1762	103	2252	5802
LS-EH-TH	315 062	1 627 822	1 911 811	217 408	1 398 652	3 051 949
	49	830	1679	103	2260	6523
LS-EH	313 949	1 569 173	1 901 077	214 738	1 372 548	2 970 702
	59	1368	4191	163	4535	19 044

A.1.1 Detailed Computational Results for Subsection 5.4.2

Table A.2: Performance of global and local phase of Algorithm 5.2, detailed numbers for Figure 5.5

# hubs	cost initial solution	cost after global phase	cost after global and local phase
2	421,253	328,658	324,584
3	381,056	325,507	318,901
4	358,917	322,412	318,289
5	343,501	320,245	317,086
6	333,869	319,965	315,814
7	328,458	319,590	314,454
8	325,493	319,339	314,531
9	324,094	319,019	313,817
10	323,150	318,814	313,141
11	322,544	318,335	313,091
12	322,041	318,671	312,464
13	321,660	318,596	311,738
14	321,376	318,423	311,363
85	320,268	318,522	309,999

Table A.3: Performance of global and local phase of Algorithm 5.2, detailed numbers for Figure 5.6

# hubs	cost initial solution	cost after global phase	cost after global and local phase
2	529,249	290,144	267,382
3	443,134	247,098	234,807
4	381,815	238,410	231,650
5	326,217	226,257	221,268
6	282,167	223,329	218,591
7	254,607	221,541	215,450
8	239,850	219,585	214,186
9	232,257	218,261	213,236
10	226,130	217,429	211,337
11	222,658	216,967	210,674
12	221,077	216,677	210,037
13	220,029	216,393	210,101
14	219,434	215,823	208,836
92	216,623	214,762	206,017

A.1.2 Detailed Computational Results for Subsection 5.4.3

Table A.4: Cost overview of incremental and non-incremental solutions on H_Auto_1, detailed numbers for Figure 5.7

# hubs	cost Alg. 1	cost Alg. 2	cost Alg. 3	cost Alg. 4	cost Alg. 5	cost Alg. 6
2	324,693	324,693	331,304	324,693	**324,584**	324,693
3	324,020	324,020	328,549	**318,756**	318,901	319,199
4	322,965	317,761	323,322	**317,695**	318,289	318,160

total cost H_Auto_1						
# hubs	cost Alg. 1	cost Alg. 2	cost Alg. 3	cost Alg. 4	cost Alg. 5	cost Alg. 6
5	320,236	316,960	**316,811**	316,969	317,086	317,104
6	320,217	316,359	**315,174**	316,359	315,814	315,557
7	314,241	315,765	**313,913**	315,765	314,454	314,536
8	**313,113**	315,765	313,464	315,765	314,531	314,520
9	313,113	315,765	**313,016**	315,765	313,817	314,520
10	313,113	315,127	**313,016**	315,127	313,141	314,520
11	**312,270**	314,983	313,016	314,983	313,091	314,520
12	**312,172**	314,983	313,016	314,983	312,464	314,417
13	312,172	314,983	313,016	314,983	**311,738**	314,268
14	312,172	314,983	313,016	314,983	**311,363**	313,893
15	**312,172**	314,983	313,016	314,983	–	313,688
16	**311,960**	314,983	313,016	314,983	–	313,688
17	**311,803**	314,983	313,016	314,983	–	313,688
18	**311,803**	314,983	312,614	314,983	–	313,688
19	311,803	314,983	**311,794**	314,983	–	313,688
20	311,803	314,983	**311,794**	314,983	–	313,688

Table A.5: Cost overview of incremental and non-incremental solutions on H_Auto_4, detailed numbers for Figure 5.8

# hubs	cost Alg. 1	cost Alg. 2	cost Alg. 3	cost Alg. 4	cost Alg. 5	cost Alg. 6
2	270,918	263,504	276,372	263,504	267,382	**247,371**
3	251,924	258,590	269,379	258,590	**234,807**	235,790
4	250,721	250,543	250,771	250,543	231,650	**226,986**
5	250,693	250,025	250,771	250,025	**221,268**	221,466
6	250,693	247,550	250,771	247,550	**218,591**	218,791
7	248,637	238,561	249,574	247,420	**215,450**	215,812
8	248,637	238,561	242,376	225,254	**214,186**	214,929
9	240,529	215,957	240,998	224,239	**213,236**	214,261
10	238,961	214,855	239,915	214,855	**211,337**	213,561
11	238,736	214,855	239,523	214,855	**210,674**	212,885
12	237,083	214,855	217,355	214,855	**210,037**	212,885
13	214,887	214,855	214,887	214,855	**210,101**	212,743
14	214,117	214,855	214,887	214,855	**208,836**	212,743
15	214,117	214,855	214,830	214,855	–	**212,743**
16	214,117	214,344	214,204	214,344	–	**212,588**
17	213,308	214,344	213,308	214,344	–	**212,275**
18	213,193	214,344	213,193	214,344	–	**212,054**
19	213,003	214,344	213,003	214,344	–	**212,054**
20	213,003	214,344	213,003	214,344	–	**212,054**

A.1.3 Detailed Computational Results for Subsection 5.4.4

Table A.6: Cost overview of of `cost-close` on H_Auto_1, detailed numbers for Figure 5.9.

# hubs	cost `greedy-ldm-ssp`	cost `cost-close`	cost `hub-mip-heur`	cost `mip-close`
2	324,693	324,815	**324,584**	324,693
3	324,020	**318,540**	318,901	319,199
4	322,965	**316,803**	318,289	318,160
5	320,236	**316,336**	317,086	317,104
6	320,217	**315,452**	315,814	315,557
7	**314,241**	314,948	314,454	314,536
8	**313,113**	314,646	314,531	314,520
9	**313,113**	313,521	313,817	314,520
10	**313,113**	313,521	313,141	314,520
11	312,270	**312,146**	313,091	314,520
12	312,172	**312,146**	312,464	314,417
13	312,172	312,146	**311,738**	314,268
14	312,172	311,762	**311,363**	313,893
15	312,172	**311,131**	–	313,688
16	311,960	**311,131**	–	313,688
17	311,803	**311,120**	–	313,688
18	311,803	**311,120**	–	313,688
19	311,803	**311,120**	–	313,688
20	311,803	**311,120**	–	313,688

Table A.7: Cost overview of of `cost-close` on H_Auto_2, detailed numbers for Figure 5.10.

# hubs	cost `greedy-ldm-ssp`	cost `cost-close`
2	**1,770,937**	1,771,719
3	**1,727,301**	1,728,885
4	1,724,299	**1,702,331**
5	1,702,189	**1,698,903**
6	1,678,979	**1,677,395**
7	**1,662,345**	1,668,994
8	**1,653,671**	1,653,677
9	1,648,171	**1,645,146**
10	**1,641,995**	1,644,494
11	1,640,164	**1,626,673**
12	1,636,475	**1,626,673**
13	1,626,391	**1,623,112**
14	1,625,780	**1,617,758**
15	1,623,209	**1,616,953**
16	1,620,428	**1,612,334**
17	1,615,472	**1,609,285**
18	1,613,356	**1,608,244**
19	1,613,071	**1,607,064**
20	1,613,071	**1,607,064**

Table A.8: Cost overview of of `cost-close` on H_Auto_3, detailed numbers for Figure 5.11.

# hubs	cost `greedy-ldm-ssp`	cost `cost-close`	cost `mip-close`
2	2,029,290	**2,015,472**	–
3	2,010,784	1,988,741	**1,984,548**
4	1,982,531	1,976,945	**1,973,973**
5	**1,955,377**	1,963,337	1,964,875
6	**1,943,243**	1,947,247	1,946,535
7	**1,939,736**	1,940,209	1,940,511
8	1,935,695	**1,932,909**	1,936,030
9	1,930,473	1,928,883	**1,928,721**
10	1,927,896	1,927,963	**1,923,535**
11	1,924,432	1,927,591	**1,917,779**
12	1,919,936	1,926,010	**1,916,625**
13	1,919,439	1,925,710	**1,913,834**
14	**1,914,098**	1,925,710	–
15	**1,910,622**	1,923,670	–
16	1,910,622	1,922,174	**1,906,911**
17	**1,906,227**	1,921,443	–
18	**1,906,127**	1,916,912	–
19	**1,903,671**	1,916,565	–
20	**1,902,851**	1,911,159	1,903,048

Table A.9: Cost overview of of `cost-close` on H_Auto_4, detailed numbers for Figure 5.12.

# hubs	cost Alg. 1	cost Alg. 2	cost Alg. 3	cost Alg. 4
2	270,918	247,465	267,382	**247,371**
3	251,924	239,073	**234,807**	235,790
4	250,721	227,027	231,650	**226,986**
5	250,693	**220,629**	221,268	221,466
6	250,693	219,375	**218,591**	218,791
7	248,637	218,628	**215,450**	215,812
8	248,637	217,682	**214,186**	214,929
9	240,529	217,482	**213,236**	214,261
10	238,961	215,390	**211,337**	213,561
11	238,736	215,347	**210,674**	212,885
12	237,083	214,369	**210,037**	212,885
13	214,887	214,049	**210,101**	212,743
14	214,117	213,260	**208,836**	212,743
15	214,117	213,260	–	**212,743**
16	214,117	213,021	–	**212,588**
17	213,308	213,021	–	**212,275**
18	213,193	213,021	–	**212,054**
19	213,003	213,021	–	**212,054**
20	213,003	213,021	–	**212,054**

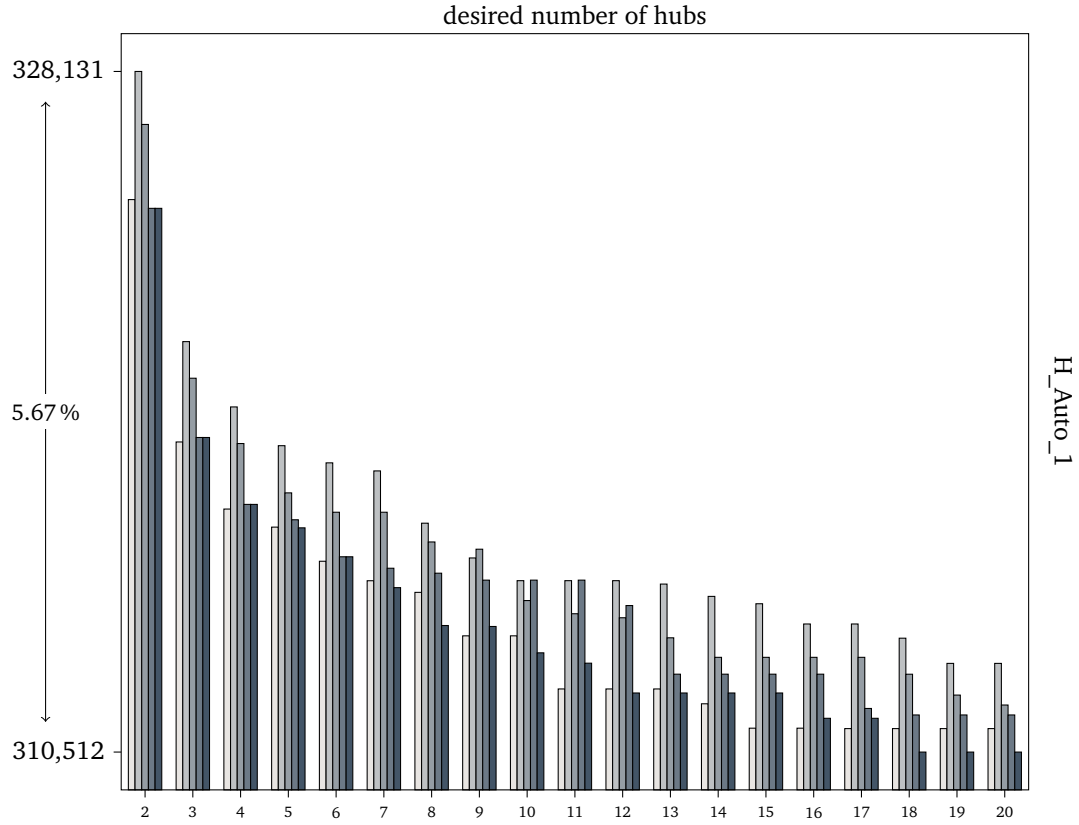
Table A.10: Cost overview of of `cost-close` on H_Auto_5, detailed numbers for Figure 5.13.

# hubs	cost `greedy-ldm-ssp`	cost `cost-close`	cost `mip-close`
2	**1,560,286**	1,578,944	–
3	**1,511,790**	1,514,024	**1,511,790**
4	1,493,951	**1,491,277**	1,492,796
5	1,486,369	1,479,293	**1,475,240**
6	1,468,563	1,457,986	**1,455,918**
7	1,460,234	1,450,470	**1,450,398**
8	1,454,889	**1,442,403**	1,442,934
9	1,434,119	1,438,651	**1,433,898**
10	**1,432,696**	1,432,953	1,432,867
11	**1,424,494**	1,426,620	1,429,945
12	**1,421,723**	1,422,754	1,425,991
13	1,419,013	**1,416,467**	1,424,230
14	1,416,168	**1,410,435**	–
15	1,409,973	**1,403,653**	–
16	1,403,510	**1,401,366**	1,416,274
17	**1,398,839**	1,399,905	–
18	**1,392,608**	1,397,411	–
19	**1,391,138**	1,394,893	–
20	**1,388,465**	1,392,726	1,403,862

A.1.4 Aggregation with `fac-loc-agg` from Subsection 5.4.5

All running times reported in this subection include the aggregation step and the time needed for the `cost-close` heuristic.

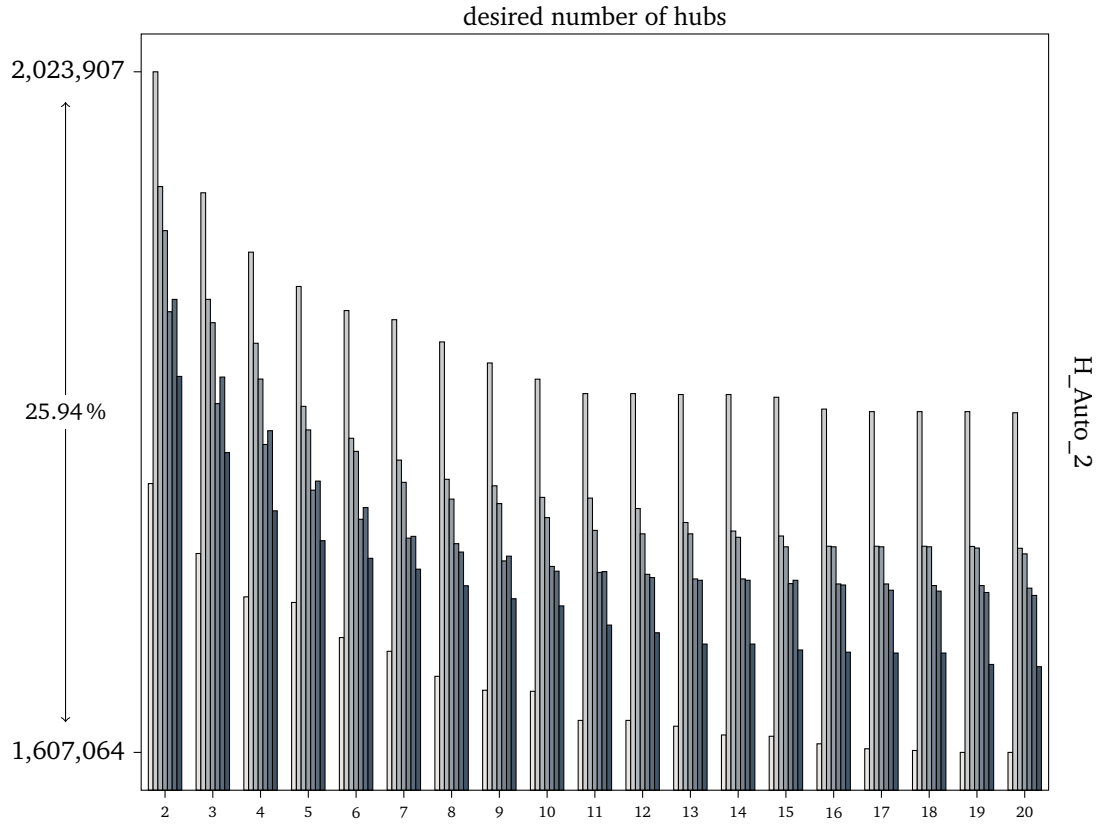
Results for H_Auto_1 to H_Auto_5



		Alg. 1	Alg. 2	Alg. 3	Alg. 4	Alg. 5
running time	[min]	11.4	8.9	7.0	11.4	6.7
maxGap	[%]	0.27	1.09	0.72	0.90	0.29
avgGap	[%]	0.06	0.85	0.54	0.31	0.05

H_Auto_1					
# hubs	cost Alg. 1	cost Alg. 2	cost Alg. 3	cost Alg. 4	cost Alg. 5
2	324,815	328,131	326,760	**324,588**	**324,588**
3	**318,540**	321,138	320,190	318,656	318,656
4	**316,803**	319,447	318,499	316,924	316,924
5	316,336	318,444	317,222	316,526	**316,317**
6	**315,452**	317,999	316,721	315,568	315,568
7	314,948	317,791	316,721	315,271	**314,769**
8	314,646	316,437	315,951	315,143	**313,788**
9	**313,521**	315,537	315,764	314,963	313,763
10	313,521	314,949	314,435	314,963	**313,081**
11	**312,146**	314,949	314,093	314,963	312,813
12	312,146	314,949	313,989	314,306	**312,041**
13	312,146	314,861	313,470	312,529	**312,041**
14	**311,762**	314,542	312,965	312,529	312,041
15	**311,131**	314,352	312,965	312,529	312,041
16	**311,131**	313,828	312,965	312,529	311,387
17	**311,120**	313,828	312,965	311,641	311,387
18	311,120	313,460	312,527	311,473	**310,512**
19	311,120	312,808	311,988	311,473	**310,512**
20	311,120	312,808	311,730	311,473	**310,512**

Aggregation with `fac-loc-agg` on H_Auto_2

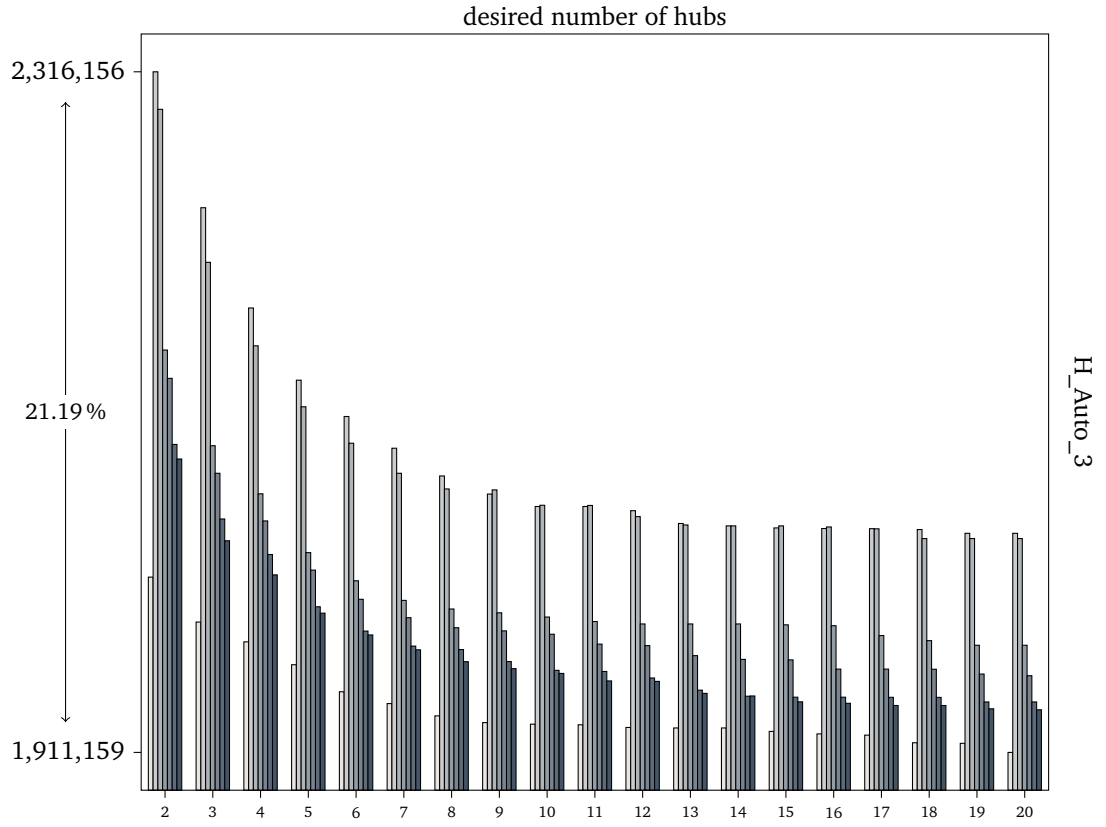


		Alg. 1	Alg. 2	Alg. 3	Alg. 4	Alg. 5	Alg. 6	Alg. 7
running time	[min]	880.8	119.5	117.5	178.5	138.5	213.0	169.0
maxGap	[%]	0.00	14.23	10.27	8.74	6.36	6.37	3.71
avgGap	[%]	0.00	12.54	7.89	7.28	5.43	5.50	3.29

	H_Auto_2						
# hubs	cost Alg. 1	cost Alg. 2	cost Alg. 3	cost Alg. 4	cost Alg. 5	cost Alg. 6	cost Alg. 7
2	**1,771,719**	2,023,907	1,953,617	1,926,623	1,876,965	1,884,520	1,837,337
3	**1,728,885**	1,949,816	1,884,551	1,870,258	1,820,705	1,836,948	1,790,676
4	**1,702,331**	1,913,447	1,857,652	1,835,727	1,795,651	1,804,084	1,755,038

H_Auto_2							
# hubs	cost Alg. 1	cost Alg. 2	cost Alg. 3	cost Alg. 4	cost Alg. 5	cost Alg. 6	cost Alg. 7
5	**1,698,903**	1,892,431	1,819,011	1,804,584	1,767,687	1,773,228	1,736,739
6	**1,677,395**	1,877,665	1,799,475	1,791,446	1,749,841	1,757,010	1,725,964
7	**1,668,994**	1,872,082	1,786,089	1,772,469	1,738,301	1,739,388	1,719,298
8	**1,653,677**	1,858,469	1,774,343	1,762,217	1,734,917	1,729,781	1,709,140
9	**1,645,146**	1,845,585	1,770,344	1,759,472	1,724,344	1,727,260	1,701,191
10	**1,644,494**	1,835,663	1,763,263	1,750,857	1,720,960	1,718,057	1,696,791
11	**1,626,673**	1,826,821	1,762,820	1,743,064	1,717,303	1,717,821	1,685,052
12	**1,626,673**	1,826,821	1,756,422	1,740,933	1,716,138	1,714,139	1,680,347
13	**1,623,112**	1,826,261	1,747,869	1,740,933	1,713,300	1,712,493	1,673,422
14	**1,617,758**	1,826,261	1,742,638	1,738,813	1,713,300	1,712,493	1,673,422
15	**1,616,953**	1,824,597	1,739,604	1,732,992	1,710,455	1,712,493	1,669,790
16	**1,612,334**	1,817,336	1,733,337	1,732,992	1,710,238	1,709,587	1,668,459
17	**1,609,285**	1,815,776	1,733,337	1,732,992	1,710,238	1,706,406	1,667,943
18	**1,608,244**	1,815,776	1,733,337	1,732,992	1,709,257	1,705,853	1,667,943
19	**1,607,064**	1,815,776	1,733,213	1,732,248	1,709,257	1,705,017	1,660,946
20	**1,607,064**	1,815,097	1,732,079	1,728,667	1,707,671	1,703,213	1,659,555

Aggregation with `fac-loc-agg` on H_Auto_3

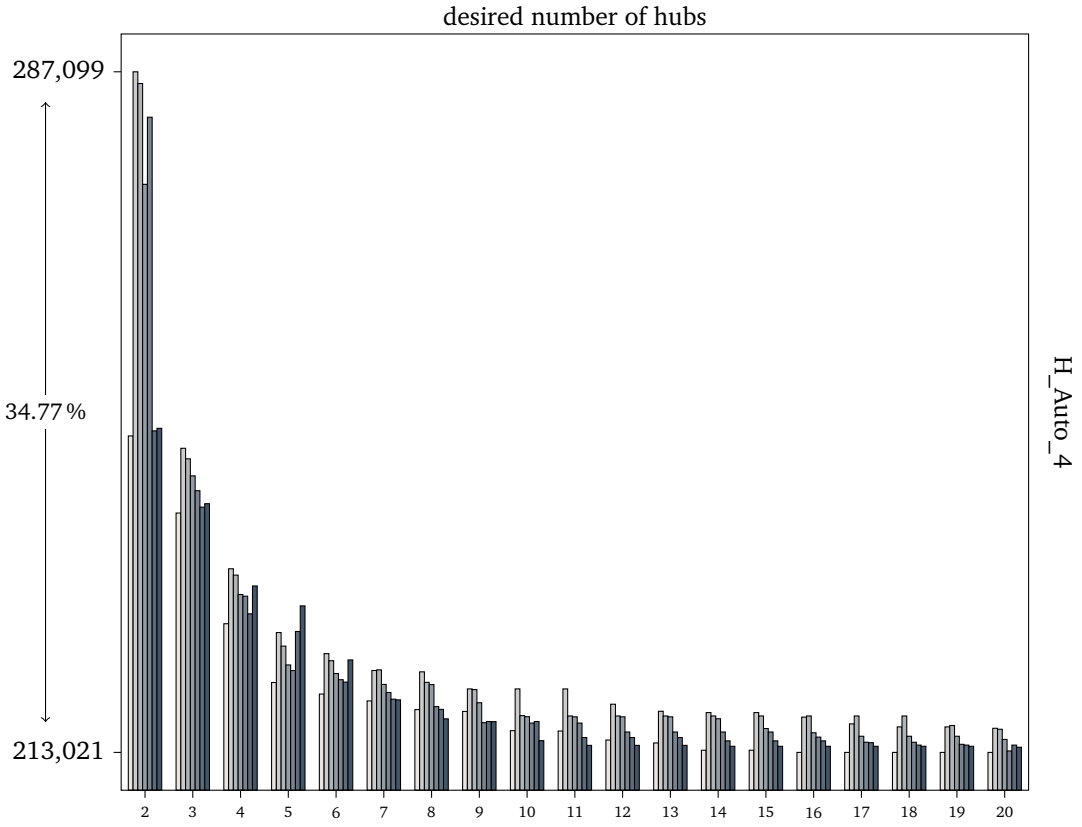


		Alg. 1	Alg. 2	Alg. 3	Alg. 4	Alg. 5	Alg. 6	Alg. 7
running time	[min]	2192.4	677.4	51.0	861.7	61.7	998.1	72.1
maxGap	[%]	0.00	14.92	13.81	6.70	5.87	3.92	3.49
avgGap	[%]	0.00	7.81	7.43	3.60	2.79	1.77	1.52

	H_Auto_3						
# hubs	cost Alg. 1	cost Alg. 2	cost Alg. 3	cost Alg. 4	cost Alg. 5	cost Alg. 6	cost Alg. 7
2	**2,015,472**	2,316,156	2,293,831	2,150,603	2,133,729	2,094,423	2,085,751
3	**1,988,741**	2,235,275	2,202,800	2,093,664	2,077,260	2,050,064	2,037,095
4	**1,976,945**	2,175,622	2,153,108	2,065,038	2,048,902	2,028,920	2,016,804

H_Auto_3							
# hubs	cost Alg. 1	cost Alg. 2	cost Alg. 3	cost Alg. 4	cost Alg. 5	cost Alg. 6	cost Alg. 7
5	**1,963,337**	2,132,666	2,116,823	2,030,035	2,019,644	1,997,821	1,994,016
6	**1,947,247**	2,111,043	2,095,151	2,013,284	2,002,276	1,983,355	1,981,056
7	**1,940,209**	2,092,171	2,077,266	2,001,623	1,991,324	1,974,412	1,972,201
8	**1,932,909**	2,075,668	2,067,933	1,996,493	1,985,351	1,972,346	1,965,139
9	**1,928,883**	2,064,888	2,067,374	1,994,229	1,983,478	1,965,246	1,960,990
10	**1,927,963**	2,057,490	2,058,233	1,991,705	1,981,484	1,959,997	1,958,162
11	**1,927,591**	2,057,490	2,058,119	1,989,044	1,975,594	1,959,330	1,953,701
12	**1,926,010**	2,054,984	2,051,451	1,987,601	1,974,699	1,955,438	1,953,427
13	**1,925,710**	2,047,384	2,046,504	1,987,601	1,968,737	1,948,209	1,946,288
14	**1,925,710**	2,045,966	2,045,958	1,987,601	1,966,525	1,944,596	1,944,743
15	**1,923,670**	2,044,747	2,045,958	1,987,076	1,966,223	1,944,009	1,941,259
16	**1,922,174**	2,044,387	2,045,313	1,986,545	1,960,732	1,944,009	1,940,391
17	**1,921,443**	2,044,285	2,044,210	1,980,677	1,960,732	1,943,946	1,939,075
18	**1,916,912**	2,043,774	2,038,407	1,977,675	1,960,646	1,943,903	1,939,075
19	**1,916,565**	2,041,535	2,038,407	1,974,920	1,957,791	1,941,185	1,937,099
20	**1,911,159**	2,041,535	2,038,407	1,974,920	1,956,796	1,941,185	1,936,519

Aggregation with `fac-loc-agg` on H_Auto_4

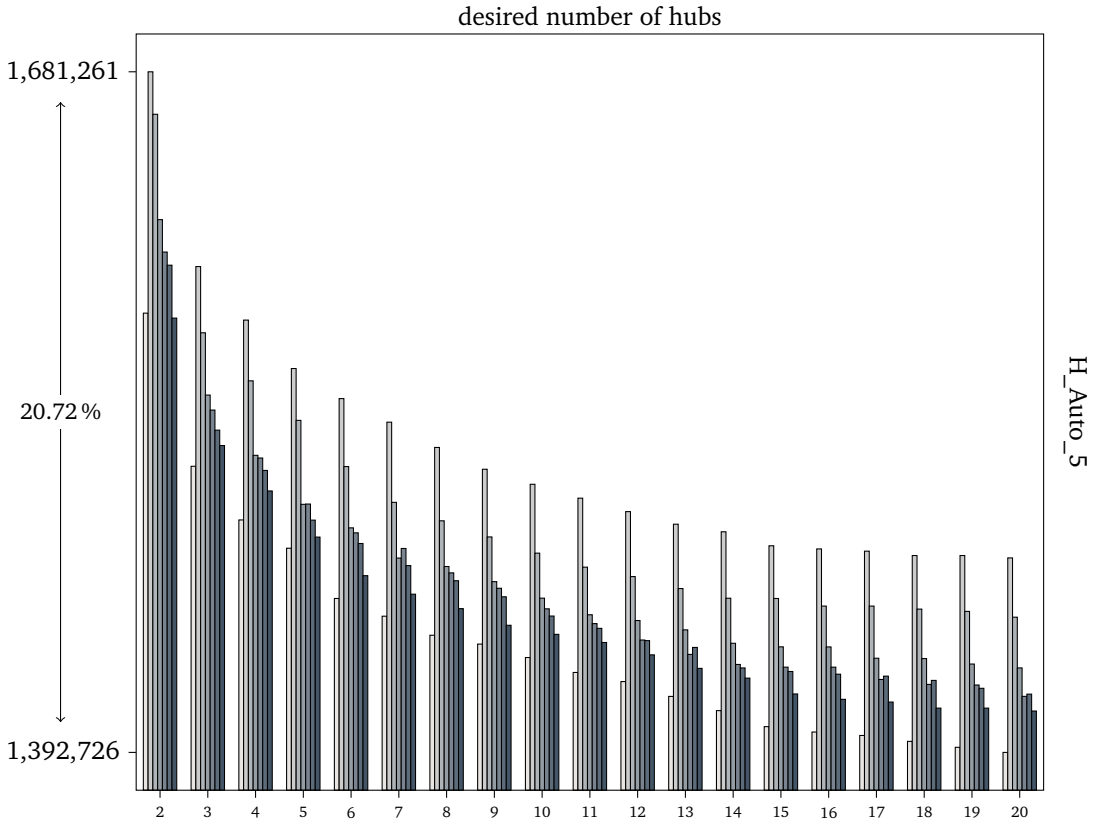


		Alg. 1	Alg. 2	Alg. 3	Alg. 4	Alg. 5	Alg. 6	Alg. 7
running time	[min]	26.8	17.3	15.1	16.8	14.5	17.5	18.1
maxGap	[%]	0.73	16.02	15.50	11.07	14.02	2.52	3.78
avgGap	[%]	0.14	2.73	2.43	1.69	1.38	0.53	0.53

	H_Auto_4						
# hubs	cost Alg. 1	cost Alg. 2	cost Alg. 3	cost Alg. 4	cost Alg. 5	cost Alg. 6	cost Alg. 7
2	**247,465**	287,099	285,824	274,851	282,155	248,018	248,295
3	**239,073**	246,123	244,976	243,121	241,507	239,716	240,091
4	**227,027**	233,007	232,320	230,207	230,023	228,103	231,138

H_Auto_4							
# hubs	cost Alg. 1	cost Alg. 2	cost Alg. 3	cost Alg. 4	cost Alg. 5	cost Alg. 6	cost Alg. 7
5	**220,629**	226,068	224,591	222,542	221,928	226,181	228,968
6	**219,375**	223,773	222,999	221,610	220,934	220,674	223,087
7	**218,628**	221,935	222,005	220,417	219,544	218,814	218,738
8	217,682	221,790	220,643	220,405	218,006	217,704	**216,676**
9	217,482	219,933	219,854	218,429	**216,258**	216,384	216,374
10	215,390	219,933	217,023	216,905	216,215	216,384	**214,297**
11	215,347	219,933	216,987	216,895	216,215	214,642	**213,790**
12	214,369	218,268	216,987	216,895	215,243	214,642	**213,790**
13	214,049	217,500	216,987	216,895	215,243	214,642	**213,790**
14	**213,260**	217,356	216,987	216,686	215,243	214,279	213,687
15	**213,260**	217,356	216,987	215,621	215,243	214,279	213,687
16	**213,021**	216,862	216,987	215,159	214,686	214,279	213,687
17	**213,021**	216,128	216,987	214,780	214,125	214,080	213,687
18	**213,021**	215,799	216,987	214,780	214,125	213,816	213,687
19	**213,021**	215,799	215,951	214,780	213,903	213,816	213,687
20	**213,021**	215,647	215,541	214,433	213,183	213,816	213,584

Aggregation with `fac-loc-agg` on H_Auto_5

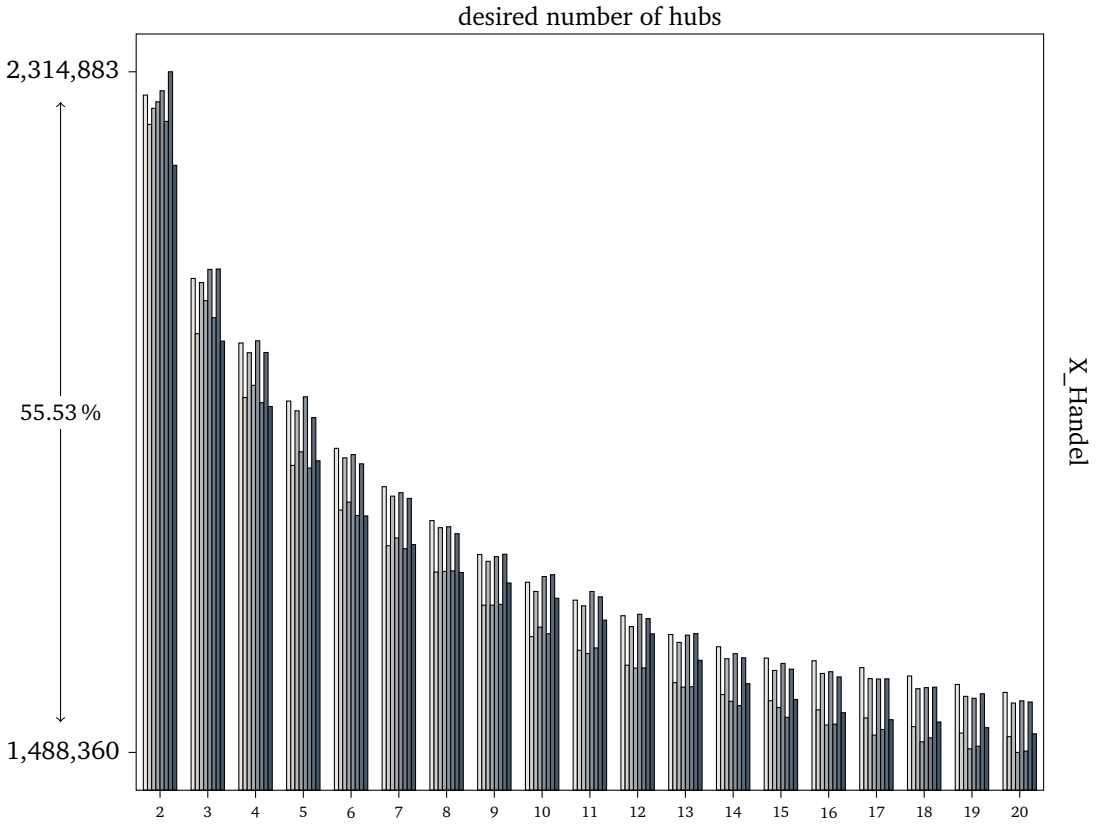


		Alg. 1	Alg. 2	Alg. 3	Alg. 4	Alg. 5	Alg. 6	Alg. 7
running time	[min]	4309.1	691.6	64.2	1152.0	74.7	1400.5	98.4
maxGap	[%]	0.13	6.62	5.48	2.65	1.98	1.85	1.26
avgGap	[%]	0.01	5.53	3.70	2.08	1.65	1.47	0.79

	H_Auto_5						
# hubs	cost Alg. 1	cost Alg. 2	cost Alg. 3	cost Alg. 4	cost Alg. 5	cost Alg. 6	cost Alg. 7
2	1,578,944	1,681,261	1,663,259	1,618,583	1,604,850	1,599,274	**1,576,852**
3	**1,514,024**	1,598,688	1,570,617	1,544,220	1,537,854	1,529,351	1,522,811
4	**1,491,277**	1,576,006	1,550,249	1,518,673	1,517,530	1,512,269	1,503,566

H_Auto_5							
# hubs	cost Alg. 1	cost Alg. 2	cost Alg. 3	cost Alg. 4	cost Alg. 5	cost Alg. 6	cost Alg. 7
5	**1,479,293**	1,555,466	1,533,492	1,497,900	1,498,022	1,491,201	1,484,013
6	**1,457,986**	1,542,714	1,513,904	1,487,924	1,485,813	1,481,285	1,467,655
7	**1,450,470**	1,532,727	1,498,733	1,475,121	1,479,181	1,471,924	1,459,802
8	**1,442,403**	1,522,009	1,490,912	1,471,539	1,468,846	1,465,508	1,453,675
9	**1,438,651**	1,512,769	1,484,066	1,465,117	1,462,308	1,458,698	1,446,623
10	**1,432,953**	1,506,395	1,477,199	1,458,123	1,453,591	1,450,541	1,442,776
11	**1,426,620**	1,500,510	1,471,263	1,451,079	1,447,333	1,445,322	1,439,327
12	**1,422,754**	1,494,808	1,467,266	1,448,634	1,440,378	1,440,111	1,434,083
13	**1,416,467**	1,489,476	1,462,191	1,444,681	1,434,273	1,437,233	1,428,356
14	**1,410,435**	1,486,261	1,458,097	1,438,987	1,430,015	1,428,536	1,424,221
15	**1,403,653**	1,480,326	1,457,984	1,437,477	1,428,871	1,427,035	1,417,500
16	**1,401,366**	1,478,987	1,454,769	1,437,444	1,428,871	1,425,889	1,415,235
17	**1,399,905**	1,478,047	1,454,769	1,432,648	1,423,657	1,425,063	1,414,076
18	**1,397,411**	1,476,185	1,453,484	1,432,510	1,421,557	1,423,253	1,411,505
19	**1,394,893**	1,476,185	1,452,504	1,430,195	1,421,293	1,419,909	1,411,505
20	**1,392,726**	1,475,182	1,450,052	1,428,544	1,416,485	1,417,434	1,410,272

Aggregation with `fac-loc-agg` on X_Handel

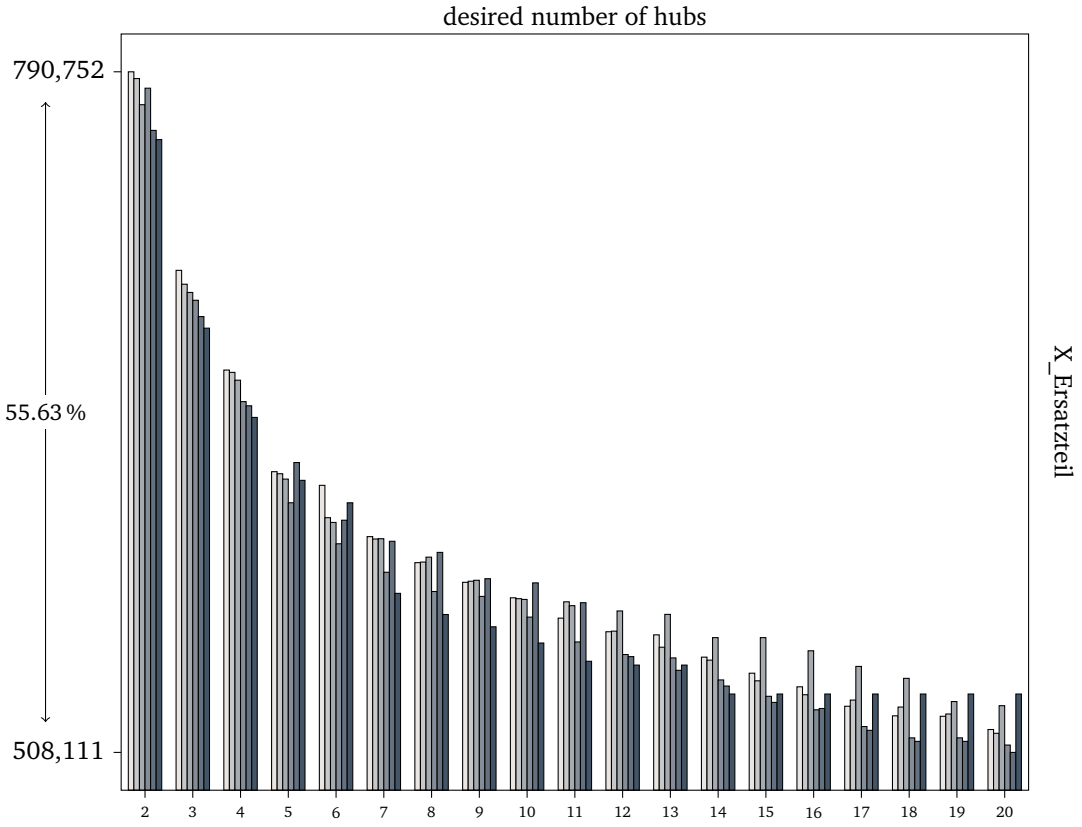


		Alg. 1	Alg. 2	Alg. 3	Alg. 4	Alg. 5	Alg. 6	Alg. 7	Alg. 8
running time	[min]	828.7	928.9	971.9	2272.8	938.8	1138.9	831.5	1041.3
maxGap	[%]	5.43	2.26	4.54	3.50	4.71	2.42	5.16	2.86
avgGap	[%]	4.42	0.71	3.70	0.63	4.18	0.32	3.98	1.19

	X_Handel							
# hubs	cost Alg. 1	cost Alg. 2	cost Alg. 3	cost Alg. 4	cost Alg. 5	cost Alg. 6	cost Alg. 7	cost Alg. 8
2	2,286,639	2,250,978	2,270,548	2,278,422	2,291,868	2,254,675	2,314,883	**2,201,333**
3	2,063,943	1,996,854	2,058,880	2,036,981	2,074,876	2,016,264	2,075,410	**1,987,860**

X_Handel								
# hubs	cost Alg. 1	cost Alg. 2	cost Alg. 3	cost Alg. 4	cost Alg. 5	cost Alg. 6	cost Alg. 7	cost Alg. 8
4	1,985,581	1,919,340	1,973,732	1,934,190	1,988,186	1,913,063	1,974,000	**1,908,444**
5	1,915,042	1,836,925	1,903,208	1,853,407	1,920,182	**1,833,726**	1,894,980	1,842,452
6	1,857,579	1,782,675	1,846,061	1,792,320	1,849,992	1,776,011	1,838,788	**1,775,509**
7	1,811,027	1,739,341	1,799,549	1,748,833	1,803,754	**1,735,801**	1,796,846	1,740,796
8	1,769,818	1,707,390	1,761,263	1,708,185	1,762,374	1,708,811	1,753,944	**1,706,870**
9	1,728,618	1,667,160	1,720,411	**1,667,101**	1,726,057	1,668,148	1,729,000	1,694,079
10	1,694,962	**1,628,993**	1,683,890	1,640,490	1,701,969	1,632,496	1,704,130	1,675,562
11	1,673,288	1,612,421	1,666,362	**1,608,458**	1,683,823	1,615,238	1,677,241	1,649,041
12	1,654,367	1,594,275	1,641,128	**1,590,719**	1,656,152	1,591,036	1,650,876	1,632,350
13	1,631,546	1,573,062	1,621,955	**1,567,709**	1,630,776	1,568,137	1,632,608	1,600,245
14	1,616,663	1,558,623	1,602,081	1,550,496	1,608,253	**1,545,083**	1,603,236	1,571,746
15	1,602,941	1,551,117	1,587,912	1,542,895	1,596,382	**1,530,990**	1,589,492	1,552,656
16	1,599,653	1,540,077	1,584,270	**1,521,596**	1,586,381	1,522,735	1,580,025	1,536,567
17	1,591,323	1,530,246	1,577,993	**1,509,404**	1,577,456	1,516,163	1,577,718	1,528,096
18	1,581,134	1,519,700	1,565,720	**1,501,274**	1,567,053	1,506,052	1,567,634	1,525,290
19	1,570,872	1,511,905	1,556,428	**1,492,735**	1,554,118	1,495,867	1,559,542	1,518,377
20	1,561,237	1,507,453	1,548,349	**1,488,360**	1,550,995	1,489,939	1,549,568	1,510,964

Aggregation with `fac-loc-agg` on X_Ersatzteil



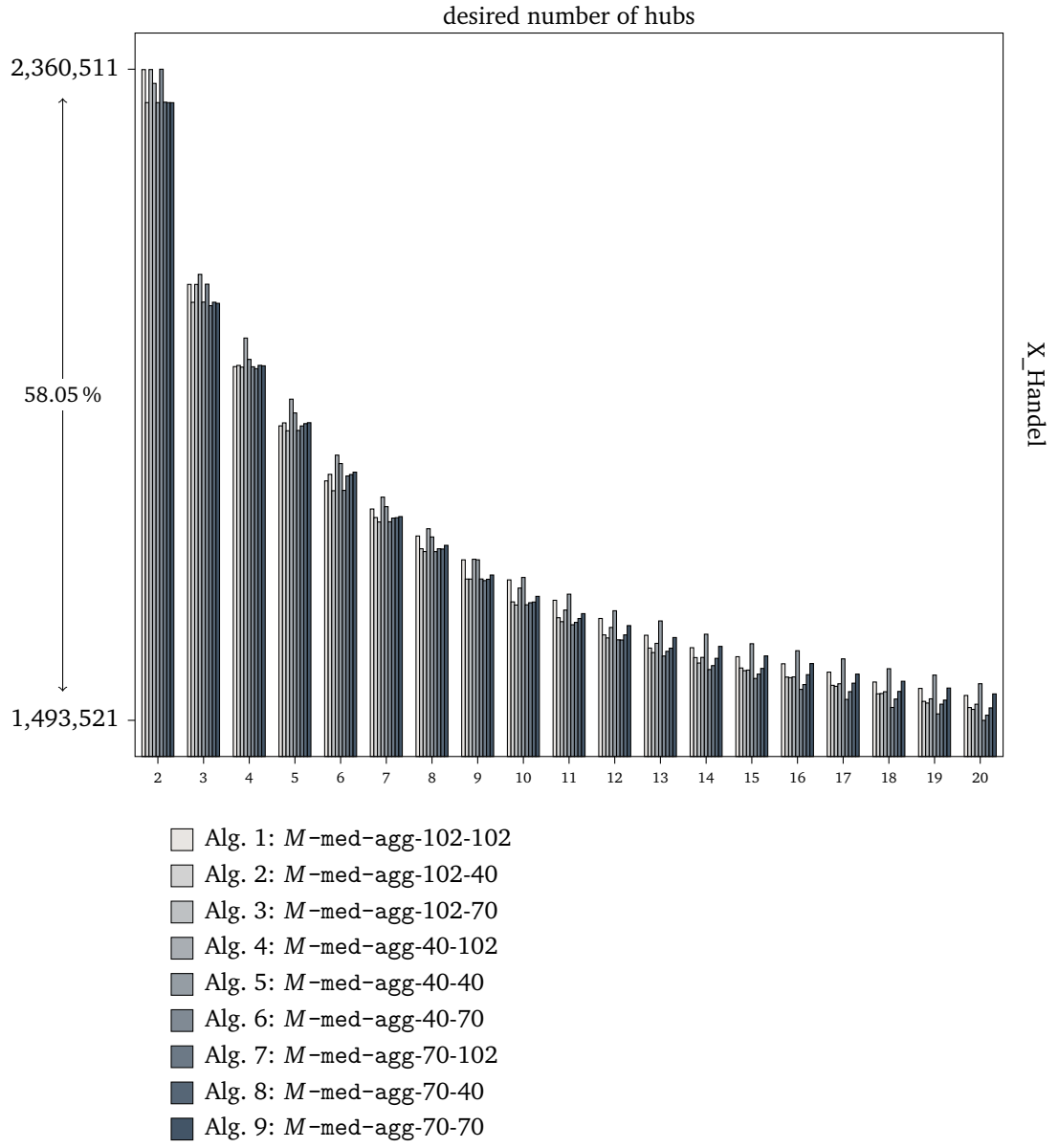
		Alg. 1	Alg. 2	Alg. 3	Alg. 4	Alg. 5	Alg. 6
running time	[min]	2939.7	4962.4	6228.3	4831.6	7386.1	10419.2
maxGap	[%]	4.10	4.52	5.13	2.80	4.56	4.77
avgGap	[%]	2.87	2.66	3.61	1.01	1.50	1.16

	X_Ersatzteil					
# hubs	cost Alg. 1	cost Alg. 2	cost Alg. 3	cost Alg. 4	cost Alg. 5	cost Alg. 6
2	790,752	787,938	777,091	783,948	766,418	**762,591**
3	708,307	702,535	699,122	695,871	689,098	**684,292**
4	666,889	665,927	662,685	653,778	652,055	**647,222**
5	624,690	623,839	621,615	**611,781**	628,486	621,093

X_Ersatzteil						
# hubs	cost Alg. 1	cost Alg. 2	cost Alg. 3	cost Alg. 4	cost Alg. 5	cost Alg. 6
6	619,015	605,590	603,617	**594,755**	604,537	611,773
7	597,719	596,739	596,860	582,923	595,789	**574,162**
8	586,909	587,160	589,200	574,934	591,181	**565,378**
9	578,754	579,198	579,593	572,900	580,247	**560,272**
10	572,309	571,943	571,643	564,293	578,519	**553,539**
11	563,873	570,655	569,049	553,980	570,276	**545,956**
12	558,224	558,465	566,853	548,758	547,890	**544,373**
13	556,925	551,792	565,430	547,350	**542,209**	544,373
14	547,674	546,431	555,774	538,204	535,679	**532,355**
15	541,003	537,847	555,774	531,411	**528,877**	532,355
16	535,353	532,065	550,317	**525,812**	526,349	532,355
17	527,285	529,841	543,815	518,847	**517,278**	532,355
18	523,303	526,948	538,866	514,155	**512,695**	532,355
19	523,112	524,063	529,233	514,155	**512,695**	532,355
20	517,621	516,051	527,514	511,165	**508,111**	532,355

A.1.5 Aggregation with M-med-agg from Subsection 5.4.6

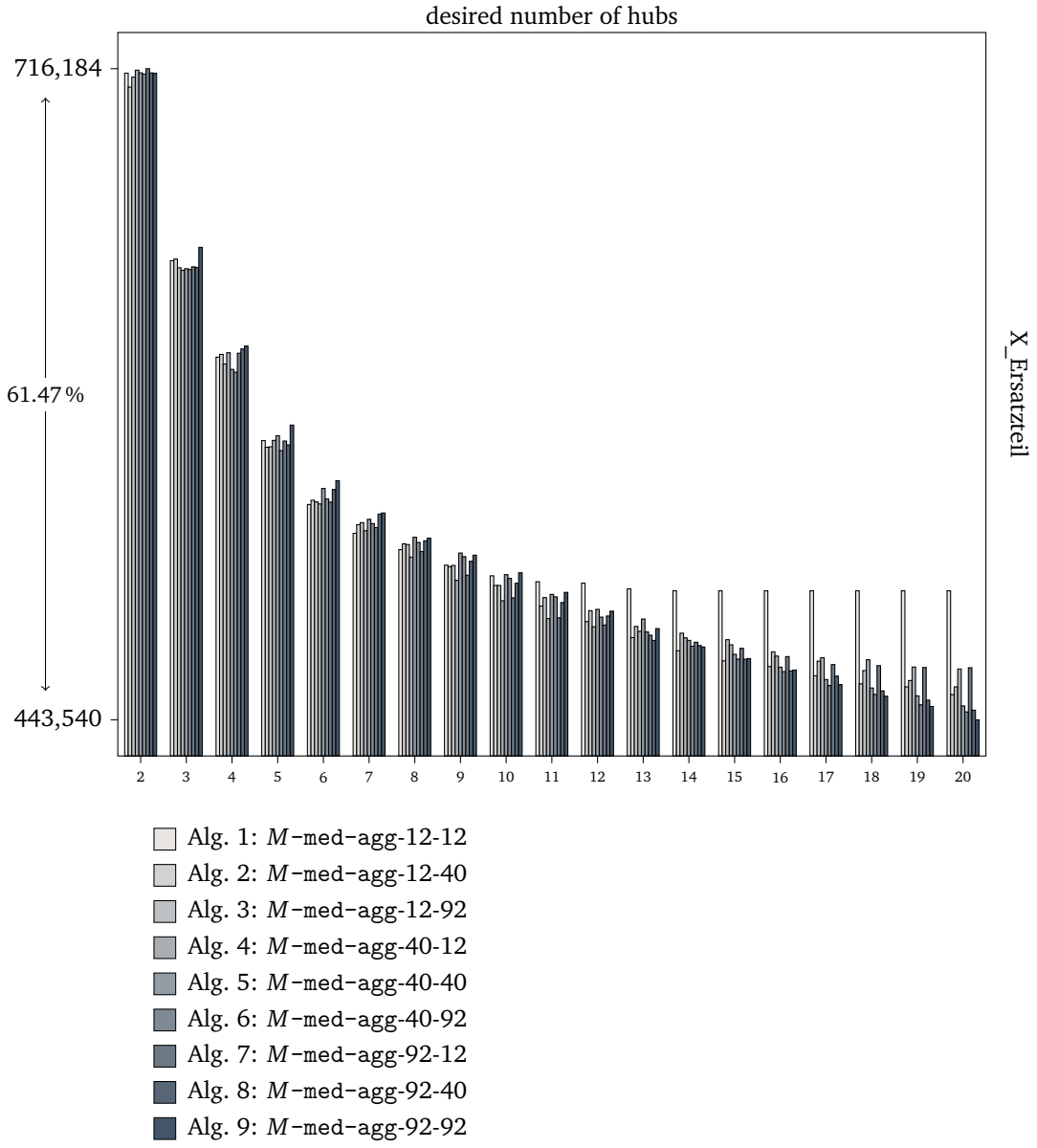
Aggregation with M-med-agg on X_Handel



		Alg. 1	Alg. 2	Alg. 3	Alg. 4	Alg. 5	Alg. 6	Alg. 7	Alg. 8	Alg. 9
running time	[min]	234.7	112.7	109.6	584.9	61.5	111.3	235.1	65.1	101.0
maxGap	[%]	2.39	1.23	1.91	2.63	3.55	1.92	1.09	1.42	2.36
avgGap	[%]	1.63	0.66	0.57	1.50	2.17	0.19	0.35	0.68	1.23

X_Handel									
# hubs	cost Alg. 1	cost Alg. 2	cost Alg. 3	cost Alg. 4	cost Alg. 5	cost Alg. 6	cost Alg. 7	cost Alg. 8	cost Alg. 9
2	2,360,027	2,316,035	2,360,073	2,341,615	2,316,035	2,360,511	2,316,608	2,316,035	**2,315,952**
3	2,073,992	2,050,151	2,073,972	2,087,367	2,050,452	2,074,295	**2,045,681**	2,050,229	2,048,795
4	1,964,531	1,966,122	1,963,844	2,002,530	1,974,139	1,964,296	**1,961,766**	1,966,200	1,965,598
5	1,885,635	1,889,630	**1,879,099**	1,921,208	1,902,816	1,879,536	1,885,526	1,888,604	1,889,863
6	1,812,366	1,821,096	**1,799,271**	1,846,629	1,835,413	1,799,728	1,818,933	1,820,865	1,823,990
7	1,774,949	1,763,572	**1,757,625**	1,790,741	1,778,045	1,757,909	1,762,850	1,763,361	1,764,841
8	1,738,979	1,721,945	1,718,232	1,748,659	1,737,738	**1,718,137**	1,721,978	1,721,638	1,726,567
9	1,707,212	1,681,646	1,681,605	1,708,035	1,707,246	1,681,729	**1,679,637**	1,681,378	1,687,073
10	1,680,527	1,651,252	**1,647,042**	1,669,650	1,683,797	1,647,387	1,650,173	1,651,077	1,658,649
11	1,653,395	1,630,183	1,624,920	1,640,385	1,661,537	**1,620,827**	1,623,899	1,629,121	1,635,597
12	1,629,164	1,607,273	1,603,265	1,617,213	1,639,424	1,600,706	**1,600,486**	1,607,306	1,619,691
13	1,606,828	1,589,719	1,583,768	1,596,018	1,625,893	**1,579,162**	1,585,415	1,589,584	1,603,788
14	1,590,260	1,577,071	1,569,856	1,577,341	1,608,189	**1,561,158**	1,566,394	1,576,092	1,591,990
15	1,578,244	1,563,239	1,559,709	1,560,453	1,595,558	**1,549,315**	1,555,080	1,562,631	1,579,320
16	1,568,864	1,551,149	1,550,328	1,551,394	1,586,218	**1,534,733**	1,541,133	1,554,355	1,569,163
17	1,557,716	1,540,150	1,538,817	1,542,224	1,575,387	**1,521,391**	1,531,723	1,543,031	1,555,083
18	1,544,543	1,528,821	1,529,260	1,531,632	1,562,277	**1,510,698**	1,521,871	1,531,992	1,545,570
19	1,536,058	1,518,798	1,516,563	1,522,162	1,553,794	**1,502,090**	1,515,047	1,520,441	1,536,530
20	1,526,672	1,510,590	1,508,128	1,514,997	1,542,281	**1,493,521**	1,500,363	1,510,127	1,528,716

Aggregation with `fac-loc-agg` on X_Ersatzteil



		Alg. 1	Alg. 2	Alg. 3	Alg. 4	Alg. 5	Alg. 6	Alg. 7	Alg. 8	Alg. 9
running time	[min]	48.0	32.5	54.1	63.4	59.4	136.7	63.8	101.5	250.3
maxGap	[%]	12.18	2.38	3.11	4.79	2.27	1.98	4.92	1.65	2.37
avgGap	[%]	4.16	0.80	1.37	1.19	1.14	0.64	1.18	0.84	1.11

	X_Ersatzteil								
# hubs	cost Alg. 1	cost Alg. 2	cost Alg. 3	cost Alg. 4	cost Alg. 5	cost Alg. 6	cost Alg. 7	cost Alg. 8	cost Alg. 9
2	714,294	**708,495**	712,647	715,573	714,464	713,903	716,184	714,413	714,272

	X_Ersatzteil								
# hubs	cost Alg. 1	cost Alg. 2	cost Alg. 3	cost Alg. 4	cost Alg. 5	cost Alg. 6	cost Alg. 7	cost Alg. 8	cost Alg. 9
3	635,791	636,467	632,828	**631,747**	632,438	632,107	633,218	632,931	641,338
4	595,375	596,556	592,510	597,245	590,270	**589,117**	597,049	598,842	600,034
5	560,435	557,685	557,851	560,537	562,500	**556,302**	560,277	558,656	566,949
6	**533,691**	535,464	534,782	533,920	540,393	536,009	534,700	539,946	543,647
7	**521,632**	525,236	526,088	522,719	527,490	525,688	524,025	529,677	530,138
8	514,775	517,223	516,919	**511,668**	519,983	517,806	513,960	518,482	519,590
9	508,278	507,624	508,195	**501,949**	513,363	511,866	504,098	509,910	512,480
10	503,847	499,805	499,790	**493,396**	504,323	502,752	494,586	500,719	505,103
11	501,365	491,141	494,722	**485,953**	496,042	495,028	486,265	492,639	496,903
12	500,782	484,686	489,230	**482,507**	489,789	486,539	483,233	487,006	489,025
13	498,371	477,978	482,635	480,590	485,751	480,332	478,960	**476,772**	481,741
14	497,548	**472,464**	479,815	477,855	476,810	474,372	476,019	474,626	473,995
15	497,548	**468,221**	477,090	474,958	470,967	468,970	473,455	468,908	469,185
16	497,548	465,842	471,992	470,230	465,583	**463,738**	470,007	464,030	464,360
17	497,548	461,975	468,120	469,559	460,387	**457,912**	466,679	461,825	458,304
18	497,548	458,621	464,160	468,695	456,861	454,192	466,219	455,658	**453,414**
19	497,548	457,339	459,998	465,646	453,560	449,819	465,434	451,783	**449,132**
20	497,548	454,100	457,340	464,805	449,350	446,845	465,355	447,583	**443,540**

A.1.6 Detailed Computational Results for Subsection 5.4.7

Table A.20: Cost overview of of aggregation techniques on X_Handel, detailed numbers for Figure 5.14.

# hubs	cost Alg. 1	cost Alg. 2	cost Alg. 3	cost Alg. 4	cost Alg. 5	cost Alg. 6
2	2,288,294	**2,151,915**	2,270,548	2,254,675	2,316,035	2,360,511
3	2,104,067	2,070,781	2,058,880	**2,016,264**	2,050,151	2,074,295
4	2,069,066	2,016,787	1,973,732	**1,913,063**	1,966,122	1,964,296
5	1,853,836	**1,799,338**	1,903,208	1,833,726	1,889,630	1,879,536
6	1,810,796	**1,724,423**	1,846,061	1,776,011	1,821,096	1,799,728
7	1,740,060	**1,710,353**	1,799,549	1,735,801	1,763,572	1,757,909
8	1,689,635	**1,670,462**	1,761,263	1,708,811	1,721,945	1,718,137
9	**1,657,253**	1,665,565	1,720,411	1,668,148	1,681,646	1,681,729
10	**1,631,405**	1,643,480	1,683,890	1,632,496	1,651,252	1,647,387
11	**1,593,447**	1,595,298	1,666,362	1,615,238	1,630,183	1,620,827
12	**1,574,244**	1,587,134	1,641,128	1,591,036	1,607,273	1,600,706
13	1,569,749	**1,564,421**	1,621,955	1,568,137	1,589,719	1,579,162
14	1,565,832	1,558,271	1,602,081	**1,545,083**	1,577,071	1,561,158
15	1,557,597	1,551,948	1,587,912	**1,530,990**	1,563,239	1,549,315
16	**1,517,699**	1,547,567	1,584,270	1,522,735	1,551,149	1,534,733
17	**1,510,846**	1,547,254	1,577,993	1,516,163	1,540,150	1,521,391
18	**1,495,606**	1,540,053	1,565,720	1,506,052	1,528,821	1,510,698
19	**1,487,994**	1,538,660	1,556,428	1,495,867	1,518,798	1,502,090
20	**1,486,394**	1,534,904	1,548,349	1,489,939	1,510,590	1,493,521

Table A.21: Cost overview of of aggregation techniques on X_Ersatzteil, detailed numbers for Figure 5.15

# hubs	cost Alg. 1	cost Alg. 2	cost Alg. 3	cost Alg. 4
2	**713,903**	714,413	783,948	766,418
3	**632,107**	632,931	695,871	689,098
4	**589,117**	598,842	653,778	652,055
5	**556,302**	558,656	611,781	628,486
6	**536,009**	539,946	594,755	604,537
7	**525,688**	529,677	582,923	595,789
8	**517,806**	518,482	574,934	591,181
9	511,866	**509,910**	572,900	580,247
10	502,752	**500,719**	564,293	578,519
11	495,028	**492,639**	553,980	570,276
12	**486,539**	487,006	548,758	547,890
13	480,332	**476,772**	547,350	542,209
14	**474,372**	474,626	538,204	535,679
15	468,970	**468,908**	531,411	528,877
16	**463,738**	464,030	525,812	526,349
17	**457,912**	461,825	518,847	517,278
18	**454,192**	455,658	514,155	512,695
19	**449,819**	451,783	514,155	512,695
20	**446,845**	447,583	511,165	508,111

A.2 Detailed Computational Results for Cost Robust Hub Location

A.2.1 Evaluation for H_Auto_1

Table A.22: Cost Robust Hub Location on H_Auto_1 and Uncertainty Set LB_avg-UB_0

opt. for	avg hist. cost	wc-sc. g=0%	wc-sc. g=5%	wc-sc. g=10%	wc-sc. g=20%	wc-sc. g=40%
best avg. scen.	**339 114** (0.0)	**314 810** (0.0)	366 975 (0.8)	406 780 (1.0)	477 178 (2.4)	571 201 (2.3)
sol. g=0%	340 080 (0.3)	315 895 (0.3)	**364 166** (0.0)	406 874 (1.1)	477 274 (2.4)	568 218 (1.8)
sol. g=5%	341 643 (0.7)	317 880 (1.0)	364 682 (0.1)	407 878 (1.3)	476 537 (2.3)	567 011 (1.6)
sol. g=10%	341 822 (0.8)	320 755 (1.9)	365 243 (0.3)	**402 574** (0.0)	**465 908** (0.0)	558 559 (0.1)
sol. g=20%	343 270 (1.2)	323 136 (2.6)	367 621 (0.9)	405 741 (0.8)	468 643 (0.6)	558 971 (0.1)
sol. g=40%	343 484 (1.3)	325 132 (3.3)	369 126 (1.4)	406 514 (1.0)	468 464 (0.5)	**558 173** (0.0)

Table A.23: Cost Robust Hub Location on H_Auto_1 and Uncertainty Set LB_avg-UB_1

opt. for	avg hist. cost	wc-sc. g=0%	wc-sc. g=5%	wc-sc. g=10%	wc-sc. g=20%	wc-sc. g=40%
best avg. scen.	**339 114** (0.0)	**314 810** (0.0)	366 935 (0.5)	406 288 (0.9)	475 779 (2.1)	569 739 (2.5)
sol. g=0%	342 606 (1.0)	318 052 (1.0)	366 474 (0.4)	408 528 (1.5)	478 257 (2.6)	569 599 (2.5)
sol. g=5%	341 624 (0.7)	317 801 (1.0)	365 331 (0.1)	407 435 (1.2)	476 910 (2.3)	567 458 (2.1)
sol. g=10%	341 813 (0.8)	320 757 (1.9)	**365 112** (0.0)	**402 476** (0.0)	**466 198** (0.0)	556 162 (0.1)
sol. g=20%	342 781 (1.1)	322 531 (2.5)	366 866 (0.5)	404 096 (0.4)	466 549 (0.1)	558 469 (0.5)
sol. g=40%	344 457 (1.6)	327 026 (3.9)	371 784 (1.8)	407 615 (1.3)	469 628 (0.7)	**555 585** (0.0)

Table A.24: Cost Robust Hub Location on H_Auto_1 and Uncertainty Set LB_avg-UB_4

opt. for	avg hist. cost	wc-sc. g=0%	wc-sc. g=5%	wc-sc. g=10%	wc-sc. g=20%	wc-sc. g=40%
best avg. scen.	339 114 (0.1)	**314 810** (0.0)	366 447 (0.4)	406 167 (1.2)	473 450 (2.4)	557 593 (2.3)
sol. g=0%	**338 881** (0.0)	315 557 (0.2)	**364 981** (0.0)	405 437 (1.0)	469 129 (1.5)	553 637 (1.6)
sol. g=5%	341 259 (0.7)	317 685 (0.9)	365 329 (0.1)	407 028 (1.4)	471 854 (2.1)	555 027 (1.8)
sol. g=10%	341 653 (0.8)	321 241 (2.0)	365 621 (0.2)	**401 514** (0.0)	**462 186** (0.0)	548 210 (0.6)
sol. g=20%	342 966 (1.2)	322 982 (2.6)	367 243 (0.6)	403 020 (0.4)	462 677 (0.1)	548 125 (0.6)
sol. g=40%	344 974 (1.8)	328 470 (4.3)	373 062 (2.2)	408 791 (1.8)	466 724 (1.0)	**544 966** (0.0)

Table A.25: Cost Robust Hub Location on H_Auto_1 and Uncertainty Set LB_1-UB_0

opt. for	avg hist. cost	wc-sc. g=0%	wc-sc. g=5%	wc-sc. g=10%	wc-sc. g=20%	wc-sc. g=40%
best avg. scen.	**339 114** (0.0)	313 751 (6.8)	359 042 (6.3)	396 170 (5.7)	452 625 (4.0)	534 337 (3.8)
sol. g=0%	349 899 (3.2)	**293 782** (0.0)	**337 823** (0.0)	**374 635** (0.0)	437 676 (0.6)	532 572 (3.5)
sol. g=5%	351 655 (3.7)	297 085 (1.1)	341 384 (1.1)	377 996 (0.9)	438 968 (0.9)	529 814 (2.9)
sol. g=10%	351 067 (3.5)	300 338 (2.2)	344 692 (2.0)	378 762 (1.1)	436 642 (0.3)	529 475 (2.9)
sol. g=20%	349 037 (2.9)	307 109 (4.5)	350 774 (3.8)	384 416 (2.6)	**435 222** (0.0)	517 539 (0.5)
sol. g=40%	349 300 (3.0)	310 452 (5.7)	354 717 (5.0)	387 768 (3.5)	436 313 (0.3)	**514 758** (0.0)

Table A.26: Cost Robust Hub Location on H_Auto_1 and Uncertainty Set LB_1-UB_1

opt. for	avg hist. cost	wc-sc. g=0%	wc-sc. g=5%	wc-sc. g=10%	wc-sc. g=20%	wc-sc. g=40%
best avg. scen.	**339 114** (0.0)	313 751 (6.0)	359 593 (5.3)	396 398 (4.8)	451 608 (4.2)	531 308 (3.5)
sol. g=0%	351 065 (3.5)	**296 065** (0.0)	341 462 (0.0)	378 602 (0.1)	438 170 (1.1)	528 770 (3.0)
sol. g=5%	352 806 (4.0)	296 826 (0.3)	**341 370** (0.0)	378 391 (0.1)	442 700 (2.2)	532 136 (3.7)
sol. g=10%	351 038 (3.5)	300 326 (1.4)	344 324 (0.9)	**378 096** (0.0)	436 935 (0.9)	528 535 (3.0)
sol. g=20%	349 462 (3.1)	303 127 (2.4)	347 319 (1.7)	380 130 (0.5)	**433 218** (0.0)	516 239 (0.6)
sol. g=40%	348 277 (2.7)	309 123 (4.4)	353 125 (3.4)	385 419 (1.9)	434 686 (0.3)	**513 214** (0.0)

Table A.27: Cost Robust Hub Location on H_Auto_1 and Uncertainty Set LB_1-UB_4

opt. for	avg hist. cost	wc-sc. g=0%	wc-sc. g=5%	wc-sc. g=10%	wc-sc. g=20%	wc-sc. g=40%
best avg. scen.	**339 114** (0.0)	313 751 (5.9)	359 440 (5.3)	393 265 (4.7)	446 689 (5.2)	519 914 (3.5)
sol. g=0%	351 079 (3.5)	**296 388** (0.0)	**341 415** (0.0)	**375 610** (0.0)	433 119 (2.0)	515 541 (2.6)
sol. g=5%	351 693 (3.7)	297 882 (0.5)	342 489 (0.3)	376 413 (0.2)	434 458 (2.3)	517 281 (2.9)
sol. g=10%	351 259 (3.6)	299 664 (1.1)	344 287 (0.8)	375 922 (0.1)	433 967 (2.2)	516 347 (2.7)
sol. g=20%	345 006 (1.7)	299 886 (1.2)	342 063 (0.2)	375 785 (0.0)	**424 555** (0.0)	502 772 (0.0)
sol. g=40%	344 114 (1.5)	307 899 (3.9)	351 802 (3.0)	383 144 (2.0)	432 548 (1.9)	**502 537** (0.0)

Table A.28: Cost Robust Hub Location on H_Auto_1 and Uncertainty Set LB_4-UB_0

opt. for	avg hist. cost	wc-sc. g=0%	wc-sc. g=5%	wc-sc. g=10%	wc-sc. g=20%	wc-sc. g=40%
best avg. scen.	**339 114** (0.0)	408 564 (3.4)	453 048 (3.5)	487 068 (3.7)	537 016 (3.6)	603 323 (3.1)
sol. g=0%	345 080 (1.8)	395 678 (0.2)	438 529 (0.1)	470 455 (0.1)	522 484 (0.8)	592 113 (1.2)
sol. g=5%	343 406 (1.3)	**395 063** (0.0)	**437 874** (0.0)	470 800 (0.2)	**518 538** (0.0)	590 967 (1.0)
sol. g=10%	344 283 (1.5)	395 995 (0.2)	439 115 (0.3)	**469 916** (0.0)	519 454 (0.2)	590 198 (0.9)
sol. g=20%	346 086 (2.1)	399 047 (1.0)	442 561 (1.1)	472 540 (0.6)	519 655 (0.2)	590 365 (0.9)
sol. g=40%	345 505 (1.9)	405 442 (2.6)	448 089 (2.3)	478 316 (1.8)	522 849 (0.8)	**584 943** (0.0)

Table A.29: Cost Robust Hub Location on H_Auto_1 and Uncertainty Set LB_4-UB_1

opt. for	avg hist. cost	wc-sc. g=0%	wc-sc. g=5%	wc-sc. g=10%	wc-sc. g=20%	wc-sc. g=40%
best avg. scen.	**339 114** (0.0)	408 564 (3.6)	452 834 (3.7)	486 013 (3.7)	536 064 (3.4)	601 316 (3.1)
sol. g=0%	341 844 (0.8)	**394 195** (0.0)	**436 511** (0.0)	**468 659** (0.0)	518 917 (0.1)	589 046 (1.0)
sol. g=5%	343 362 (1.3)	395 050 (0.2)	437 551 (0.2)	469 412 (0.2)	**518 233** (0.0)	589 348 (1.0)
sol. g=10%	343 954 (1.4)	395 635 (0.4)	437 551 (0.2)	470 218 (0.3)	519 009 (0.1)	588 973 (1.0)
sol. g=20%	345 593 (1.9)	398 508 (1.1)	440 748 (1.0)	472 188 (0.8)	518 247 (0.0)	588 389 (0.9)
sol. g=40%	344 104 (1.5)	407 230 (3.3)	449 358 (2.9)	479 280 (2.3)	525 140 (1.3)	**583 318** (0.0)

Table A.30: Cost Robust Hub Location on H_Auto_1 and Uncertainty Set LB_4-UB_4

opt. for	avg hist. cost	wc-sc. g=0%	wc-sc. g=5%	wc-sc. g=10%	wc-sc. g=20%	wc-sc. g=40%
best avg. scen.	**339 114** (0.0)	408 564 (3.3)	450 677 (3.3)	483 283 (3.9)	530 234 (4.0)	586 699 (3.5)
sol. g=0%	345 061 (1.8)	395 552 (0.0)	436 698 (0.1)	468 162 (0.6)	515 964 (1.2)	576 677 (1.7)
sol. g=5%	342 786 (1.1)	396 162 (0.2)	437 254 (0.3)	467 285 (0.5)	512 854 (0.6)	575 139 (1.4)
sol. g=10%	342 739 (1.1)	**395 510** (0.0)	**436 162** (0.0)	**465 169** (0.0)	**509 736** (0.0)	573 074 (1.0)
sol. g=20%	344 057 (1.5)	398 967 (0.9)	440 881 (1.1)	467 857 (0.6)	512 888 (0.6)	570 707 (0.6)
sol. g=40%	344 850 (1.7)	406 856 (2.9)	448 565 (2.8)	475 256 (2.2)	517 283 (1.5)	**567 125** (0.0)

A.2.2 Evaluation for H_Auto_4

Table A.31: Cost Robust Hub Location on H_Auto_4 and Uncertainty Set LB_avg-UB_0

opt. for	avg hist. cost	wc-sc. g=0%	wc-sc. g=5%	wc-sc. g=10%	wc-sc. g=20%	wc-sc. g=40%
best avg. scen.	**208 120** (0.0)	**218 846** (0.0)	257 260 (0.4)	281 914 (0.9)	323 534 (1.7)	389 796 (4.0)
sol. g=0%	209 252 (0.5)	224 233 (2.5)	262 552 (2.4)	286 445 (2.5)	326 568 (2.7)	388 110 (3.5)
sol. g=5%	209 318 (0.6)	221 440 (1.2)	**256 285** (0.0)	279 571 (0.0)	320 174 (0.7)	381 365 (1.7)
sol. g=10%	210 457 (1.1)	224 586 (2.6)	258 038 (0.7)	**279 456** (0.0)	**317 990** (0.0)	377 865 (0.8)
sol. g=20%	212 117 (1.9)	225 731 (3.1)	259 151 (1.1)	280 904 (0.5)	318 782 (0.2)	378 302 (0.9)
sol. g=40%	210 885 (1.3)	227 041 (3.7)	258 969 (1.0)	279 972 (0.2)	318 401 (0.1)	**374 981** (0.0)

Table A.32: Cost Robust Hub Location on H_Auto_4 and Uncertainty Set LB_avg-UB_1

opt. for	avg hist. cost	wc-sc. g=0%	wc-sc. g=5%	wc-sc. g=10%	wc-sc. g=20%	wc-sc. g=40%
best avg. scen.	**208 120** (0.0)	**218 846** (0.0)	257 202 (0.4)	281 634 (1.1)	323 039 (2.2)	387 651 (4.3)
sol. g=0%	209 155 (0.5)	224 110 (2.4)	262 654 (2.5)	285 638 (2.5)	325 242 (2.9)	384 335 (3.4)
sol. g=5%	209 382 (0.6)	222 014 (1.4)	256 763 (0.2)	279 368 (0.3)	318 272 (0.7)	377 756 (1.6)
sol. g=10%	210 679 (1.2)	223 099 (1.9)	**256 174** (0.0)	278 832 (0.1)	316 859 (0.2)	374 891 (0.9)
sol. g=20%	210 043 (0.9)	224 846 (2.7)	257 883 (0.7)	**278 658** (0.0)	**316 179** (0.0)	372 991 (0.3)
sol. g=40%	210 938 (1.4)	227 142 (3.8)	259 759 (1.4)	280 394 (0.6)	317 099 (0.3)	**371 713** (0.0)

Table A.33: Cost Robust Hub Location on H_Auto_4 and Uncertainty Set LB_avg-UB_4

opt. for	avg hist. cost	wc-sc. g=0%	wc-sc. g=5%	wc-sc. g=10%	wc-sc. g=20%	wc-sc. g=40%
best avg. scen.	208 120 (0.0)	**218 854** (0.0)	257 098 (0.9)	281 458 (1.4)	322 370 (2.4)	378 662 (4.0)
sol. g=0%	208 620 (0.3)	222 700 (1.8)	261 046 (2.5)	284 532 (2.5)	323 693 (2.9)	375 792 (3.3)
sol. g=5%	**208 040** (0.0)	220 262 (0.6)	**254 757** (0.0)	**277 626** (0.0)	316 626 (0.6)	369 331 (1.5)
sol. g=10%	209 317 (0.6)	224 092 (2.4)	256 763 (0.8)	278 711 (0.4)	316 181 (0.5)	367 465 (1.0)
sol. g=20%	210 083 (1.0)	225 860 (3.2)	257 870 (1.2)	278 990 (0.5)	**314 697** (0.0)	364 961 (0.3)
sol. g=40%	210 831 (1.3)	228 009 (4.2)	260 412 (2.2)	281 275 (1.3)	316 080 (0.4)	**363 944** (0.0)

Table A.34: Cost Robust Hub Location on H_Auto_4 and Uncertainty Set LB_1-UB_0

opt. for	avg hist. cost	wc-sc. g=0%	wc-sc. g=5%	wc-sc. g=10%	wc-sc. g=20%	wc-sc. g=40%
best avg. scen.	**208 120** (0.0)	212 706 (11.1)	245 630 (10.9)	267 248 (9.6)	306 303 (7.9)	369 199 (7.0)
sol. g=0%	215 609 (3.6)	**191 498** (0.0)	222 899 (0.6)	246 539 (1.1)	288 740 (1.7)	357 238 (3.6)
sol. g=5%	215 432 (3.5)	191 867 (0.2)	**221 581** (0.0)	**243 866** (0.0)	286 114 (0.8)	352 726 (2.3)
sol. g=10%	215 484 (3.5)	193 193 (0.9)	222 633 (0.5)	244 708 (0.3)	287 091 (1.1)	353 464 (2.5)
sol. g=20%	214 784 (3.2)	198 391 (3.6)	226 275 (2.1)	246 806 (1.2)	**283 835** (0.0)	345 897 (0.3)
sol. g=40%	214 013 (2.8)	202 689 (5.8)	230 799 (4.2)	251 093 (3.0)	286 987 (1.1)	**344 887** (0.0)

Table A.35: Cost Robust Hub Location on H_Auto_4 and Uncertainty Set LB_1-UB_1

opt. for	avg hist. cost	wc-sc. g=0%	wc-sc. g=5%	wc-sc. g=10%	wc-sc. g=20%	wc-sc. g=40%
best avg. scen.	**208 120** (0.0)	212 706 (11.5)	245 558 (11.3)	267 346 (9.8)	305 154 (8.1)	365 753 (7.6)
sol. g=0%	214 828 (3.2)	**190 808** (0.0)	222 387 (0.8)	246 232 (1.2)	288 907 (2.3)	355 247 (4.5)
sol. g=5%	215 396 (3.5)	191 486 (0.4)	**220 710** (0.0)	**243 429** (0.0)	285 097 (1.0)	351 569 (3.5)
sol. g=10%	215 915 (3.7)	193 343 (1.3)	222 023 (0.6)	244 207 (0.3)	284 865 (0.9)	350 741 (3.2)
sol. g=20%	214 969 (3.3)	198 921 (4.3)	226 799 (2.8)	246 574 (1.3)	**282 334** (0.0)	345 156 (1.6)
sol. g=40%	215 184 (3.4)	205 937 (7.9)	233 532 (5.8)	252 659 (3.8)	287 827 (1.9)	**339 790** (0.0)

Table A.36: Cost Robust Hub Location on H_Auto_4 and Uncertainty Set LB_1-UB_4

opt. for	avg hist. cost	wc-sc. g=0%	wc-sc. g=5%	wc-sc. g=10%	wc-sc. g=20%	wc-sc. g=40%
best avg. scen.	**208 120** (0.0)	212 706 (11.1)	245 379 (10.8)	266 848 (9.6)	303 731 (8.0)	355 444 (7.0)
sol. g=0%	214 986 (3.3)	**191 511** (0.0)	223 059 (0.7)	245 267 (0.8)	286 704 (1.9)	349 128 (5.1)
sol. g=5%	215 344 (3.5)	192 320 (0.4)	**221 521** (0.0)	**243 416** (0.0)	282 948 (0.6)	346 069 (4.1)
sol. g=10%	216 733 (4.1)	194 365 (1.5)	223 190 (0.8)	245 142 (0.7)	284 385 (1.1)	346 989 (4.4)
sol. g=20%	213 536 (2.6)	195 503 (2.1)	224 464 (1.3)	244 606 (0.5)	**281 296** (0.0)	341 617 (2.8)
sol. g=40%	214 447 (3.0)	206 167 (7.7)	233 757 (5.5)	252 672 (3.8)	286 961 (2.0)	**332 289** (0.0)

Table A.37: Cost Robust Hub Location on H_Auto_4 and Uncertainty Set LB_4-UB_0

opt. for	avg hist. cost	wc-sc. g=0%	wc-sc. g=5%	wc-sc. g=10%	wc-sc. g=20%	wc-sc. g=40%
best avg. scen.	**208 120** (0.0)	269 133 (8.2)	298 047 (8.3)	318 584 (7.2)	354 924 (6.5)	409 673 (6.5)
sol. g=0%	211 438 (1.6)	**248 807** (0.0)	275 256 (0.1)	297 487 (0.1)	335 294 (0.6)	393 450 (2.3)
sol. g=5%	213 475 (2.6)	250 798 (0.8)	276 232 (0.4)	298 590 (0.4)	337 013 (1.1)	396 301 (3.0)
sol. g=10%	212 124 (1.9)	250 510 (0.7)	**275 110** (0.0)	**297 304** (0.0)	334 697 (0.4)	391 742 (1.8)
sol. g=20%	213 680 (2.7)	255 205 (2.6)	279 337 (1.5)	298 323 (0.3)	**333 408** (0.0)	390 469 (1.5)
sol. g=40%	215 336 (3.5)	260 573 (4.7)	284 545 (3.4)	303 318 (2.0)	336 891 (1.0)	**384 695** (0.0)

Table A.38: Cost Robust Hub Location on H_Auto_4 and Uncertainty Set LB_4-UB_1

opt. for	avg hist. cost	wc-sc. g=0%	wc-sc. g=5%	wc-sc. g=10%	wc-sc. g=20%	wc-sc. g=40%
best avg. scen.	**208 120** (0.0)	269 133 (8.4)	297 689 (8.3)	317 205 (7.4)	353 597 (6.8)	403 998 (6.6)
sol. g=0%	210 699 (1.2)	**248 381** (0.0)	**274 759** (0.0)	297 186 (0.7)	332 503 (0.4)	389 605 (2.8)
sol. g=5%	211 836 (1.8)	250 063 (0.7)	276 265 (0.5)	297 630 (0.8)	334 093 (0.9)	390 635 (3.1)
sol. g=10%	211 934 (1.8)	252 715 (1.7)	277 296 (0.9)	**295 254** (0.0)	331 153 (0.0)	386 201 (1.9)
sol. g=20%	213 080 (2.4)	255 398 (2.8)	279 124 (1.6)	298 250 (1.0)	**331 087** (0.0)	386 956 (2.1)
sol. g=40%	213 966 (2.8)	258 091 (3.9)	282 145 (2.7)	300 335 (1.7)	332 896 (0.5)	**378 954** (0.0)

Table A.39: Cost Robust Hub Location on H_Auto_4 and Uncertainty Set LB_4-UB_4

opt. for	avg hist. cost	wc-sc. g=0%	wc-sc. g=5%	wc-sc. g=10%	wc-sc. g=20%	wc-sc. g=40%
best avg. scen.	**208 120** (0.0)	269 133 (8.4)	297 318 (8.0)	315 690 (6.7)	350 075 (7.2)	393 104 (6.1)
sol. g=0%	210 680 (1.2)	**248 353** (0.0)	**275 335** (0.0)	296 831 (0.4)	329 348 (0.9)	380 457 (2.7)
sol. g=5%	212 530 (2.1)	250 009 (0.7)	276 711 (0.5)	298 526 (0.9)	331 865 (1.7)	382 898 (3.3)
sol. g=10%	212 991 (2.3)	253 803 (2.2)	277 347 (0.7)	**295 773** (0.0)	**326 456** (0.0)	377 476 (1.9)
sol. g=20%	214 130 (2.9)	255 347 (2.8)	278 664 (1.2)	297 717 (0.7)	326 960 (0.2)	376 285 (1.6)
sol. g=40%	213 982 (2.8)	257 158 (3.5)	280 113 (1.7)	298 670 (1.0)	328 124 (0.5)	**370 512** (0.0)

Appendix B

Full Models

In this appendix we list some models that we used in this thesis with all inequalities written out. We also give dual variables in brackets and for each inequality its original number as a reference.

B.1 Full Models for Section 5.2

HUBLINEDGE =

$$\min \quad \sum_{n\in N} o_n g_n + \sum_{k\in K} o_k y_k + \sum_{k\in K} m_k \ell_k + \sum_{j\in J}\sum_{e\in E} hand_j(e) x_j(e) \qquad \text{(5.20 rev)}$$

$$\text{s.t.} \quad \sum_{j\in J} \sigma_{kj\pi} d_j z_{kj} + s_{k\pi} = \ell_k \qquad \forall\, k,\pi \qquad [ZL_{k\pi} \in \mathbb{R}] \qquad \text{(4.28 rev)}$$

$$y_k \geq z_{kj} \qquad \forall\, k\in K, j\in J \qquad [ZY_{kj} \geq 0] \qquad \text{(4.24 rev)}$$

$$g_n \geq \sum_{k\in K(e)} y_k \qquad \forall\, n\in N,\ e\in\delta(n) \qquad [YG_{ne} \geq 0] \qquad \text{(5.9 rev)}$$

$$\sum_{n\in N} g_n \leq M \qquad [\kappa \leq 0] \qquad \text{(5.10 rev)}$$

$$x_j(e) \leq \sum_{k\in K(e)} z_{kj} \qquad \forall\, e,j \qquad [RZ_{ej} \leq 0] \qquad \text{(5.21 rev)}$$

$$\sum_{e\in\delta^+(\text{source}_j)} x_j(e) \geq 1 \qquad \forall j \qquad [R_j \geq 0] \qquad \text{(5.22 rev)}$$

$$\sum_{e\in\delta^+(n)} x_j(e) - \sum_{e\in\delta^-(n)} x_j(e) \geq 0 \qquad \forall j\ \forall n\in N \qquad [B_{jn} \geq 0] \qquad \text{(5.23 rev)}$$

$$x_j(e), z_{kj}, y_k, g_n,\ s_{k\pi} \geq 0,\ \ell_k \in \mathbb{R} \qquad \text{(5.24 rev)}$$

HUBLINEDGEDUAL =

$$\max \quad \sum_{j\in J} R_j - \kappa M \qquad \text{(5.12 rev)}$$

$$RZ_{e(k)j} \leq \sum_{\pi\in\Pi} \sigma_{kj\pi} d_j ZL_{k\pi} + ZY_{kj} \qquad \forall k,\ j \qquad [z_{kj} \geq 0] \qquad \text{(5.14 rev)}$$

$$\sum_{j\in J} ZY_{kj} \leq \sum_{n\in N(k)} YG_{ne(k)} + o_k \qquad \forall k \qquad [y_k \geq 0] \qquad \text{(5.15 rev)}$$

$$\sum_{\pi\in\Pi} ZL_{k\pi} = m_k \qquad \forall k \qquad [\ell_k \in \mathbb{R}] \qquad \text{(5.16 rev)}$$

$$0 \leq ZL_{k\pi} \qquad \forall k,\ \pi \qquad [s_{k\pi} \geq 0] \qquad (5.17\text{ rev})$$

$$\sum_{e:n\in\delta(e)} YG_{ne} \leq \kappa + o_n \qquad \forall n \in N \qquad [g_n \geq 0] \qquad (5.18\text{ rev})$$

$$R_j - B_{j\,\mathrm{end}(e)} \leq hand_j(e) + RZ_{ej} \qquad \forall j,\ \forall e \in \delta^+(\mathrm{source}_j) \qquad [x_j(e) \geq 0] \qquad (5.25\text{ rev})$$

$$B_{j\,\mathrm{start}(e)} - B_{j\,\mathrm{end}(e)} \leq hand_j(e) + RZ_{ej} \qquad \forall j,\ \forall e \in E \cap (N \times N) \qquad [x_j(e) \geq 0] \qquad (5.26\text{ rev})$$

$$B_{j\,\mathrm{start}(e)} \leq hand_j(e) + RZ_{ej} \qquad \forall j,\ \forall e \in \delta^-(\mathrm{sink}_j) \qquad [x_j(e) \geq 0] \qquad (5.27\text{ rev})$$

$$R_j,\ \kappa, RZ_{ej},\ ZY_{kj},\ YG_{ne} \geq 0,\ ZL_{k\pi} \in \mathbb{R} \qquad (5.19\text{ rev})$$

$$B_{jn} \geq 0 \qquad (5.28\text{ rev})$$

VIOLATION =

$$\min \quad \sum_{n\in N} I_n \tag{5.29 rev}$$

$$\text{s.t.} \quad RZ_{e(k)j} \le \sum_{\pi\in\Pi} \sigma_{kj\pi} d_j ZL_{k\pi} + ZY_{kj}$$
$$\forall j,\ \forall e \in E \setminus \big((F_j)\times(F_j)\big),\ \forall k \in K(e) \quad [z_{kj} \ge 0] \tag{5.14 rev}$$

$$\sum_{\pi\in\Pi} ZL_{k\pi} = m_k \qquad \forall k : \exists (5.14) \text{ for } k \quad [\ell_k \in \mathbb{R}] \tag{5.16 rev}$$

$$0 \le ZL_{k\pi} \quad \forall k : \exists (5.14) \text{ for } k, \forall\, \pi \quad [s_{k\pi} \ge 0] \tag{5.17 rev}$$

$$\bar{B}_{j\,\mathrm{start}(e)} - B_{j\,\mathrm{end}(e)} \le hand_j(e) + RZ_{ej}$$
$$\forall j,\ \forall e \in E \cap \big((F_j)\times(N\setminus F_j)\big) \quad [x_j(e) \ge 0] \tag{5.30 rev}$$

$$B_{j\,\mathrm{start}(e)} - B_{j\,\mathrm{end}(e)} \le hand_j(e) + RZ_{ej}$$
$$\forall j,\ \forall e \in E \cap \big((N\setminus F_j)\times(N\setminus F_j)\big) \quad [x_j(e) \ge 0] \tag{5.31 rev}$$

$$B_{j\,\mathrm{start}(e)} - \bar{B}_{j\,\mathrm{end}(e)} \le hand_j(e) + RZ_{ej}$$
$$\forall j,\ \forall e \in E \cap \big((N\setminus F_j)\times(F_j)\big) \quad [x_j(e) \ge 0] \tag{5.32 rev}$$

$$\sum_{e:n\in\delta(e)} YG_{ne} \le \bar{\kappa} + o_n + I_n \qquad \forall n \quad [g_n \ge 0] \tag{5.33 rev}$$

$$\sum_{j\in J} ZY_{kj} + \sum_{\pi\in\Pi} (ZL_{k\pi}/m_k) \sum_{j\in\bar{J}(k)} \bar{ZY}_{kj\pi} \le o_k + \sum_{n\in N(k)} YG_{ne(k)}$$
$$\forall k : \exists (5.14) \text{ for } k \quad [y_k \ge 0] \tag{5.34 rev}$$

$$\text{variable:} \quad I_n,\ RZ_{ej},\ ZY_{kj},\ YG_{ne},\ B_{jn} \ge 0,\ ZL_{k\pi} \in \mathbb{R} \tag{5.35 rev}$$

$$\text{fixed:} \quad \text{all } \bar{R}_j =: \bar{B}_{j\,\mathrm{source}_j},\ \bar{\kappa},\ \text{and } \bar{B}_{jn} \text{ for } n \in F_j \tag{5.36 rev}$$

B.2 Expanded Models for Section 5.5

ROB-HUBLINPATH =

$$\min \quad \sum_{n\in N} o_n g_n + \sum_{k\in K} o_k y_k + \sum_{k\in K} m_k \ell_k + \sum_{j\in J}\sum_{P\in\mathcal{P}_j} c_j^P r_j^P + \Gamma\omega + \sum_{j\in J}\delta_j + \sum_{k\in K}\phi_k \tag{B.1}$$

s.t.

$$\sum_{P\in\mathcal{P}_j:\ e\in P} r_j^P \le \sum_{k\in K(e)} z_{kj} \qquad \forall\, e, j \qquad [RZ_{ej}\le 0] \qquad \text{(4.27 rev)}$$

$$\sum_{j\in J}\eta_{kj\pi} z_{kj} + s_{k\pi} = \ell_k \qquad \forall\, k, \pi \qquad [ZL_{k\pi}\in\mathbb{R}] \qquad \text{(4.28 rev)}$$

$$\sum_{P\in\mathcal{P}_j} r_j^P \ge 1 \qquad \forall\, j \qquad [R_j \ge 0] \qquad \text{(4.4 rev)}$$

$$y_k \ge z_{kj} \qquad \forall\, k, j \qquad [ZY_{kj}\ge 0] \qquad \text{(4.24 rev)}$$

$$g_n \ge \sum_{k\in K(e)} y_k \qquad \forall\, n\in N,\ e\in\delta(n) \qquad [YG_{ne}\ge 0] \qquad \text{(5.10 rev)}$$

$$\sum_{n\in N} g_n \le M \qquad [\kappa \le 0] \qquad \text{(5.10 rev)}$$

$$G_j\omega + \delta_j - \sum_{k\in K}\gamma_{kj} \ge \sum_{P\in\mathcal{P}_j} \tilde{c}_j^P r_j^P \qquad \forall\, j \qquad [\mu_j \ge 0] \qquad \text{(4.43 rev)}$$

$$\phi_k - \sum_{j\in J}\xi_{kj\pi} \ge -m_k s_{k\pi} \qquad \forall k,\ \pi \qquad [\theta_{k\pi}\ge 0] \qquad \text{(4.44 rev)}$$

$$\gamma_{kj} + \xi_{kj\pi} \ge m_k \tilde{\eta}_{kj\pi} z_{kj} \qquad \forall k,\ j,\ \pi \qquad [\chi_{kj\pi}\ge 0] \qquad \text{(4.45 rev)}$$

$$\omega,\ \delta_j,\ \phi_k,\ \gamma_{kj},\ \xi_{kj\pi} \ge 0 \tag{B.2}$$

$$r_j^P \in \{0,1\},\ \ z_{kj}\in\{0,1\},\ \ y_k\in\{0,1\}, g_n\in\{0,1\}\ \ \ell_k\ge 0, s_{k\pi}\in\mathbb{R} \tag{B.3}$$

The dual of ROB-HUBLINPATH reads ROB-HUBLINPATHDUAL =

$$\max \quad \sum_{j\in J} R_j - \kappa M \tag{5.12 rev}$$

$$R_j \le \tilde{c}_j^P \mu_j + \sum_{e\in P} RZ_{ej} + c_j^P \qquad \forall j,\ P\in\mathcal{P}_j \quad [r_j^P \ge 0] \tag{5.13 rev}$$

$$RZ_{e(k)j} \le \sum_{\pi\in\Pi} \eta_{kj\pi} ZL_{k\pi} + ZY_{kj} + \sum_{\pi\in\Pi} m_k \tilde{\eta}_{kj\pi} \chi_{kj\pi} \qquad \forall k,\ j \quad [z_{kj} \ge 0] \tag{5.14 rev}$$

$$\sum_{j\in J} ZY_{kj} \le \sum_{n\in N(k)} YG_{ne(k)} + o_k \qquad \forall k \quad [y_k \ge 0] \tag{5.15 rev}$$

$$\sum_{\pi\in\Pi} ZL_{k\pi} = m_k \qquad \forall k \quad [\ell_k \in \mathbb{R}] \tag{5.16 rev}$$

$$\theta_{k\pi} m_k \le ZL_{k\pi} \qquad \forall k,\ \pi \quad [s_{k\pi} \ge 0] \tag{B.4}$$

$$\sum_{e:n\in\delta(e)} YG_{ne} - \kappa \le o_n \qquad \forall n\in N \quad [g_n \ge 0] \tag{5.18 rev}$$

$$\sum_{j\in J} G_j \mu_j \le \Gamma \qquad [\omega \ge 0] \tag{4.34 rev}$$

$$\mu_j \le 1 \qquad \forall j \quad [\delta_j \ge 0] \tag{4.38 rev}$$

$$\sum_{\pi\in\Pi} \theta_{k\pi} \le 1 \qquad \forall k \quad [\phi_k \ge 0] \tag{4.35 rev}$$

$$\sum_{\pi\in\Pi} \chi_{kj\pi} \le \mu_j \qquad \forall k,\ j \quad [\gamma_{kj} \ge 0] \tag{4.39 rev}$$

$$\chi_{kj\pi} \le \theta_{k\pi} \qquad \forall,\ k,\ j,\ \pi \quad [\xi_{kj\pi} \ge 0] \tag{4.40 rev}$$

$$R_j,\ \kappa,\ \mu_j,\ \chi_{kj\pi}, \theta_{k\pi}, RZ_{ej},\ ZY_{kj},\ YG_{ne} \ge 0,\ ZL_{k\pi} \in \mathbb{R} \tag{B.5}$$

ROB-VIOLATION =

$$\min \quad \sum_{n\in N} I_n \tag{5.29 rev}$$

$$\text{s.t.} \quad RZ_{e(k)j} \le \sum_{\pi\in\Pi} \eta_{kj\pi} ZL_{k\pi} + ZY_{kj} + \sum_{\pi\in\Pi} m_k \tilde{\eta}_{kj\pi} \chi_{kj\pi}$$
$$\forall j,\ \forall e \in E \setminus \big((F_j)\times(F_j)\big),\ \forall k \in K(e) \quad [z_{kj} \ge 0] \tag{5.57 rev}$$

$$\sum_{\pi\in\Pi} ZL_{k\pi} = m_k$$
$$\forall k : \exists (5.57) \text{ for } k \quad [\ell_k \in \mathbb{R}] \tag{5.16 rev}$$

$$\theta_{k\pi} m_k \le ZL_{k\pi}$$
$$\forall k : \exists (5.57) \text{ for } k,\ \forall\ \pi \quad [s_{k\pi} \ge 0] \tag{5.17 rev}$$

$$\bar{B}_{j\,\mathrm{start}(e)} - B_{j\,\mathrm{end}(e)} \le hand_j(e) + \widetilde{hand}_j(e)\,\bar{\mu}_j + RZ_{ej}$$
$$\forall j,\ \forall e \in E \cap \big((F_j)\times(N\setminus F_j)\big) \quad [x_j(e) \ge 0] \tag{5.30 rev}$$

$$B_{j\,\mathrm{start}(e)} - B_{j\,\mathrm{end}(e)} \le hand_j(e) + \widetilde{hand}_j(e)\,\bar{\mu}_j + RZ_{ej}$$
$$\forall j,\ \forall e \in E \cap \big((N\setminus F_j)\times(N\setminus F_j)\big) \quad [x_j(e) \ge 0] \tag{5.31 rev}$$

$$B_{j\,\mathrm{start}(e)} - \bar{B}_{j\,\mathrm{end}(e)} \le hand_j(e) + \widetilde{hand}_j(e)\,\bar{\mu}_j + RZ_{ej}$$
$$\forall j,\ \forall e \in E \cap \big((N\setminus F_j)\times(F_j)\big) \quad [x_j(e) \ge 0] \tag{5.32 rev}$$

$$\sum_{e:n\in\delta(e)} YG_{ne} \le \bar{\kappa} + o_n + I_n \qquad \forall n \quad [g_n \ge 0] \tag{5.33 rev}$$

$$\sum_{j\in J} ZY_{kj} + \sum_{\pi\in\Pi} (ZL_{k\pi}/m_k) \sum_{j\in\bar{J}(k)} \bar{ZY}_{kj\pi} \le o_k + \sum_{n\in N(k)} YG_{ne(k)} \quad \forall k \quad [y_k \ge 0] \quad \text{(5.34 rev)}$$

$$\sum_{\pi\in\Pi} \chi_{kj\pi} \le \bar{\mu}_j$$

$$\forall k, j : \exists(5.57) \text{ for } k, j \quad [\gamma \ge 0] \quad \text{(4.39 rev)}$$

$$\chi_{kj\pi} \le \theta_{k\pi}$$

$$\forall \pi, \forall k, j : \exists(5.57) \text{ for } k, j \quad [\xi_{kj\pi} \ge 0] \quad \text{(4.40 rev)}$$

$$\sum_{\pi\in\Pi} \theta_{k\pi} \le 1$$

$$\forall k : \exists(5.57) \text{ for } k \quad [\phi_k \ge 0] \quad \text{(4.35 rev)}$$

variable: $\chi_{kj\pi}, \theta_{k\pi}, I_n, RZ_{ej}, ZY_{kj}, YG_{ne}, B_{jn} \ge 0, ZL_{k\pi} \in \mathbb{R}$ (B.6)

fixed: all $\bar{R}_j =: \bar{B}_{j\,\text{source}_j}$, $\bar{\kappa}$, and $\bar{B}_{jn}$ for $n \in F_j$, $\bar{\mu}_j$ for all j (B.7)

List of Figures

List of Tables

Bibliography

[4fl17] 4flow AG. *A logistics consultancy company serving small, medium-sized and global customers from a broad spectrum of industries*. 2017. URL: http://www.4flow.de/en.html (visited on 03/01/2017).

[Abe+17] F. Abed, L. Chen, Y. Disser, M. Groß, N. Megow, J. Meißner, A. T. Richter, and R. Rischke. "Scheduling Maintenance Jobs in Networks". In: *Algorithms and Complexity*. Ed. by D. Fotakis, A. Pagourtzis, and V. T. Paschos. Cham: Springer International Publishing, 2017, pp. 19–30. ISBN: 978-3-319-57586-5. DOI: 10.1007/978-3-319-57586-5_3.

[ACN01] Y. P. Aneja, R. Chandrasekaran, and K. P. K. Nair. "Maximizing residual flow under an arc destruction". In: *Networks* 38.4 (2001), pp. 194–198. DOI: 10.1002/net.10001.

[AG86] P. Afentakis and B. Gavish. "Optimal Lot-Sizing Algorithms for Complex Product Structures". In: *Oper. Res.* 34.2 (1986), pp. 237–249.

[AGK84] P. Afentakis, B. Gavish, and U. Karmarkar. "Computationally Efficient Optimal Solutions to the Lot-Sizing Problem in Multistage Assembly Systems". In: *Man. Sci.* 30.2 (1984), pp. 222–239.

[AK08] S. Alumur and B. Y. Kara. "Network hub location problems: The state of the art". In: *European Journal of Operational Research* 190.1 (2008), pp. 1–21. ISSN: 0377-2217. DOI: http://dx.doi.org/10.1016/j.ejor.2007.06.008.

[AMS15] A. Arulselvan, O. Maurer, and M. Skutella. "An incremental algorithm for the uncapacitated facility location problem". In: *Networks* 65.4 (2015), pp. 306–311. ISSN: 1097-0037. DOI: 10.1002/net.21595.

[ASV07] P. Avella, A. Sassano, and I. Vasil'ev. "Computational study of large-scale p-median problems". In: *Mathematical Programming* 109.1 (2007), pp. 89–114.

[AYP11] A. Altin, H. Yaman, and M. C. Pinar. "The robust network loading problem under hose demand uncertainty: formulation, polyhedral analysis, and computations". In: *INFORMS Journal on Computing* 23.1 (2011), pp. 75–89.

[BDG17] A. Bernstein, Y. Disser, and M. Groß. "General Bounds for Incremental Maximization". In: *CoRR* abs/1705.10253 (2017). arXiv: 1705.10253. URL: http://arxiv.org/abs/1705.10253.

[Ben+04] A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski. "Adjustable robust solutions of uncertain linear programs". In: *Mathematical Programming* 99.2 (2004), pp. 351–376.

[BKD13] A. Bley, D. Karch, and F. D'Andreagiovanni. "WDM Fiber Replacement Scheduling". In: *Electronic Notes in Discrete Mathematics* 41 (2013), pp. 189–196. ISSN: 1571-0653. DOI: 10.1016/j.endm.2013.05.092.

[BKK15a] N. Boland, T. Kalinowski, and S. Kaur. "Scheduling arc shut downs in a network to maximize flow over time with a bounded number of jobs per time period". In: *Journal of Combinatorial Optimization* (2015), pp. 1–21. ISSN: 1573-2886. DOI: 10.1007/s10878-015-9910-x.

[BKK15b] N. Boland, T. Kalinowski, and S. Kaur. "Scheduling network maintenance jobs with release dates and deadlines to maximize total flow over time: Bounds and solution strategies". In: *Computers & Operations Research* 64 (2015), pp. 113–129. ISSN: 0305-0548. DOI: http://dx.doi.org/10.1016/j.cor.2015.05.011.

[Blu+87] D. Blumenfeld, L. Burns, C. Daganzo, M. Frick, and R. Hall. "Reducing Logistics Costs at General Motors". In: *Interfaces* 17.1 (1987), pp. 26–47.

[BNS13] D. Bertsimas, E. Nasrabadi, and S. Stiller. "Robust and Adaptive Network Flows". In: *Operations Research* 61.5 (2013), pp. 1218–1242. DOI: 10.1287/opre.2013.1200.

[Bol+14] N. Boland, T. Kalinowski, H. Waterer, and L. Zheng. "Scheduling arc maintenance jobs in a network to maximize total flow over time". In: *Discrete Applied Mathematics* 163 (2014), pp. 34–52. DOI: 10.1016/j.dam.2012.05.027.

[BS03] D. Bertsimas and M. Sim. "Robust discrete optimization and network flows". English. In: *Mathematical Programming* 98 (1-3 2003), pp. 49–71. ISSN: 0025-5610. DOI: 10.1007/s10107-003-0396-4.

[BS12] N. L. Boland and M. W. P. Savelsbergh. "Optimizing the Hunter Valley Coal Chain". In: *Supply Chain Disruptions: Theory and Practice of Managing Risk*. Ed. by H. Gurnani, A. Mehrotra, and S. Ray. Springer, London, 2012, pp. 275–302. ISBN: 978-0-85729-778-5. DOI: 10.1007/978-0-85729-778-5_10.

[Bur+85] L. Burns, R. Hall, D. Blumenfeld, and C. Daganzo. "Distribution Strategies that Minimize Transportation and Inventory Costs". In: *Oper. Res.* 33.3 (1985), pp. 469–490.

[Cak09] O. Cakir. "Benders decomposition applied to multi-commodity, multi-mode distribution planning". In: *Expert Systems with Applications* 36.4 (2009), pp. 8212–8217. ISSN: 0957-4174.

[CCG09] A. Costa, J.-F. Cordeau, and B. Gendron. "Benders, metric and cutset inequalities for multicommodity capacitated network design". In: *Comput. Optim. Appl.* 42 (3 2009), pp. 371–392.

[CCG11] M. Chouman, T. Crainic, and B. Gendron. *Commodity Representations and Cutset-Based Inequalities for Multicommodity Capacitated Fixed-Charge Network Design*. Tech. rep. CIRRELT-2011-56, Centre de recherche sur les transports. Université de Montréal, 2011.

[CEK02] J. F. Campbell, A. T. Ernst, and M. Krishnamoorthy. "Hub location problems". In: *Facility location: applications and theory* 1 (2002), pp. 373–407.

[Çet05] S. Çetinkaya. "Coordination of Inventory and Shipment Consolidation Decisions: A Review of Premises, Models, and Justification". In: *Applications of Supply Chain Management and E-Commerce Research*. Ed. by P. M. Pardalos, D. Hearn, J. Geunes, E. Akçali, H. E. Romeijn, and Z.-J. M. Shen. Vol. 92. Applied Optimization. Springer, 2005, pp. 3–51.

[CG02] T. Crainic and M. Gendreau. "Cooperative parallel tabu search for capacitated network design". In: *Journal of Heuristics* 8.6 (2002), pp. 601–627.

[CGF00] T. Crainic, M. Gendreau, and J. Farvolden. "A Simplex-Based Tabu Search Method for Capacitated Network Design". In: *INFORMS Journal on Computing* 12.3 (2000), p. 223.

[CGH04] T. Crainic, B. Gendron, and G. Hernu. "A slope scaling/Lagrangean perturbation heuristic with long-term memory for multicommodity capacitated fixed-charge network design". In: *Journal of Heuristics* 10.5 (2004), pp. 525–545.

[Cha+02] L. Chan, A. Muriel, Z.-J. Shen, D. Simchi-Levi, and C.-P. Teo. "Effective Zero-Inventory-Ordering Policies for the Single-Warehouse Multiretailer Problem with Piecewise Linear Cost Structures". English. In: *Man. Sci.* 48.11 (2002), pp. 1446–1460.

[Cha+11] D. Chakrabarty, C. Chekuri, S. Khanna, and N. Korula. "Approximability of capacitated network design". In: *Integer Programming and Combinatoral Optimization*. Ed. by O. Günlük and G. J. Woeginger. Springer, 2011, pp. 78–91.

[CI98] R. Canetti and S. Irani. "Bounding the Power of Preemption in Randomized Scheduling". In: *SIAM Journal on Computing* 27.4 (1998), pp. 993–1015. DOI: 10.1137/S0097539795283292.

[CKM14] J. Chang, S. Khuller, and K. Mukherjee. "LP rounding and combinatorial algorithms for minimizing active and busy time". In: *Proc. of the 26th SPAA*. Ed. by G. E. Blelloch and P. Sanders. ACM, New York, 2014, pp. 118–127. DOI: 10.1145/2612669.2612689.

[CKM15] J. Chang, S. Khuller, and K. Mukherjee. "Active and Busy Time Minimization". In: *Proc. of the 12th MAPSP*. 2015, pp. 247–249. URL: http://feb.kuleuven.be/mapsp2015/Proceedings%20MAPSP%202015.pdf.

[CM07] S. Chopra and P. Meindl. *Supply chain management: Strategy, planning, and operations*. Pearson Prentice-Hall, 2007.

[CML09] R. S. de Camargo, G. de Miranda Jr, and H. P. L. Luna. "Benders decomposition for hub location problems with economies of scale". In: *Transportation Science* 43.1 (2009), pp. 86–97.

[CO12] J. F. Campbell and M. E. O'Kelly. "Twenty-Five Years of Hub Location Research". In: *Transportation Science* 46.2 (2012), pp. 153–169. DOI: 10.1287/trsc.1120.0410.

[Coh+15] V. Cohen-Addad, Z. Li, C. Mathieu, and I. Milis. "Energy-Efficient Algorithms for Non-preemptive Speed-Scaling". In: *Proc. of the 12th WAOA*. Ed. by E. Bampis and O. Svensson. Vol. 8952. LNCS. Springer International Publishing, 2015. Chap. Energy-Efficient Algorithms for Non-preemptive Speed-Scaling, pp. 107–118. ISBN: 978-3-319-18263-6. DOI: 10.1007/978-3-319-18263-6_10.

[Cos05] A. Costa. “A survey on benders decomposition applied to fixed-charge network design problems”. In: *Comput. Oper. Res.* 32 (6 2005), pp. 1429–1450.

[Cra+14] T. G. Crainic, M. Hewitt, M. Toulouse, and D. M. Vu. “Service Network Design with Resource Constraints”. In: *Transportation Science* Articles in Advance 02 Jul (2014). DOI: 10.1287/trsc.2014.0525.

[Cra00] T. G. Crainic. “Service network design in freight transportation”. In: *European Journal of Operational Research* 122.2 (2000), pp. 272–288.

[CS60] A. Clark and H. Scarf. “Optimal Policies for a Multi-Echelon Inventory Problem”. In: *Man. Sci.* 6.4 (1960), pp. 475–490.

[CSV12] J. R. Correa, M. Skutella, and J. Verschae. “The Power of Preemption on Unrelated Machines and Applications to Scheduling Orders”. In: *Mathematics of Operations Research* 37.2 (2012), pp. 379–398. ISSN: 0364-765X. DOI: 10.1287/moor.1110.0520.

[Däu+17] K. Däubel, Y. Disser, M. Klimm, T. Mütze, and F. Smolny. “Distance-preserving graph contractions”. In: *CoRR* abs/1705.04544 (2017). arXiv: 1705.04544. URL: http://arxiv.org/abs/1705.04544.

[DDS05] G. Desaulniers, J. Desrosiers, and M. M. Solomon. *Column generation*. Vol. 5. Springer, 2005.

[DM17] Y. Disser and J. Matuschke. “The Complexity of Computing a Robust Flow”. In: *CoRR* abs/1704.08241 (2017). arXiv: 1704.08241. URL: http://arxiv.org/abs/1704.08241.

[Ere+12] A. Erera, M. Hewitt, M. Savelsbergh, and Y. Zhang. “Improved load plan design through integer programming based local search”. In: *Transportation Science* (2012).

[FG09] A. Frangioni and B. Gendron. “0–1 reformulations of the multicommodity capacitated network design problem”. In: *Discrete Applied Mathematics* 157.6 (2009), pp. 1229–1241.

[FG13] A. Frangioni and B. Gendron. “A stabilized structured Dantzig–Wolfe decomposition method”. In: *Mathematical Programming* 140.1 (2013), pp. 45–76. ISSN: 1436-4646. DOI: 10.1007/s10107-012-0626-8.

[Fla+10] M. Flammini, G. Monaco, L. Moscardelli, H. Shachnai, M. Shalom, T. Tamir, and S. Zaks. “Minimizing total busy time in parallel scheduling with application to optical networks”. In: *Theoretical Computer Science* 411.40–42 (2010), pp. 3553–3562. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2010.05.011.

[FSZ10] M. Fischetti, D. Salvagnin, and A. Zanette. “A note on the selection of Benders’ cuts”. In: *Math. Program.* 124 (1-2 2010), pp. 175–182. ISSN: 0025-5610.

[GCG03] I. Ghamlouche, T. Crainic, and M. Gendreau. “Cycle-based neighbourhoods for fixed-charge capacitated multicommodity network design”. In: *Oper. Res.* (2003), pp. 655–667.

[GCG04] I. Ghamlouche, T. Crainic, and M. Gendreau. “Path relinking, cycle-based neighbourhoods and capacitated multicommodity network design”. In: *Annals of Oper. Res.* 131.1 (2004), pp. 109–133.

[GG74] A. Geoffrion and G. Graves. “Multicommodity Distribution System Design by Benders Decomposition”. In: *Man. Sci.* 20.5 (1974), pp. 822–844.

[GLS12] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric algorithms and combinatorial optimization*. Vol. 2. Springer Science & Business Media, 2012.

[GP03] J. Geunes and P. Pardalos. “Network optimization in supply chain management and financial engineering: An annotated bibliography”. In: *Networks* 42.2 (2003), pp. 66–84.

[GP90] G. Guisewite and P. Pardalos. “Minimum concave-cost network flow problems: Applications, complexity, and algorithms”. In: *Annals of Oper. Res.* 25 (1 1990), pp. 75–99. ISSN: 0254-5330.

[Ha92] S. Ha. “Compile-time scheduling of dataflow program graphs with dynamic constructs”. PhD thesis. University of California, Berkeley, 1992. URL: http://www.eecs.berkeley.edu/Pubs/TechRpts/1992/ERL-92-43.pdf.

[Har+16] T. Harks, F. G. König, J. Matuschke, A. T. Richter, and J. Schulz. “An Integrated Approach to Tactical Transportation Planning in Logistics Networks”. In: *Transportation Science* 50.2 (2016), pp. 439–460. DOI: 10.1287/trsc.2014.0541.

[HNS10] M. Hewitt, G. L. Nemhauser, and M. W. Savelsbergh. “Combining exact and heuristic approaches for the capacitated fixed-charge network flow problem”. In: *INFORMS Journal on Computing* 22.2 (2010), pp. 314–325.

[Jay98] V. Jayaraman. “Transportation, facility location and inventory issues in distribution network design: An investigation”. In: *International Journal of Operations and Production Management* 18.5 (1998), pp. 471–494.

[JD07] R. Jans and Z. Degraeve. “Meta-heuristics for dynamic lot sizing: A review and comparison of solution approaches”. In: *European Journal of Operational Research* 177.3 (2007), pp. 1855–1875.

[JS12] T. Jordán and I. Schlotter. *Parameterized Complexity of Spare Capacity Allocation and the Multicost Steiner Subgraph Problem*. Tech. rep. TR-2012-16. Egerváry Research Group, Eötvös Loránd University, Budapest, 2012. URL: http://www.cs.elte.hu/egres/tr/egres-12-16.pdf.

[Kha+15] R. Khandekar, B. Schieber, H. Shachnai, and T. Tamir. “Real-time Scheduling to Minimize Machine Busy Times”. In: *Journal of Scheduling* 18.6 (2015), pp. 561–573. ISSN: 1094-6136. DOI: 10.1007/s10951-014-0411-z.

[Kha80] L. G. Khachiyan. “Polynomial algorithms in linear programming”. In: *USSR Computational Mathematics and Mathematical Physics* 20.1 (1980), pp. 53–72.

[KKS10] J. Kempkes, A. Koberstein, and L. Suhl. “A Resource Based Mixed Integer Modelling Approach for Integrated Operational Logistics Planning”. In: *Advanced Manufacturing and Sustainable Logistics*. Ed. by W. Aalst, J. Mylopoulos, M. Rosemann, M. J. Shaw, C. Szyperski, W. Dangelmaier, A. Blecken, R. Delius, and S. Klöpfer. Vol. 46. Lecture Notes in Business Information Processing. Springer, 2010, pp. 281–294.

[Kli90] J. G. Klincewicz. “Solving a freight transport problem using facility location techniques”. In: *Operations Research* 38.1 (1990), pp. 99–109.

[KMR12] F. König, J. Matuschke, and A. Richter. “Multi-Dimensional Commodity Covering for Tariff Selection in Transportation”. In: *12th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2012)*. Ed. by D. Delling and L. Liberti. Vol. 25. OASICS. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2012, pp. 58–70. ISBN: 978-3-939897-45-3.

[KMS15] T. Kalinowski, D. Matsypura, and M. W. Savelsbergh. “Incremental network design with maximum flows”. In: *European Journal of Oper. Res.* 242.1 (2015), pp. 51–62. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2014.10.003. URL: http://www.sciencedirect.com/science/article/pii/S0377221714008078.

[KP99] D. Kim and P. Pardalos. “A solution approach to the fixed charge network flow problem using a dynamic slope scaling procedure”. In: *Oper. Res. Letters* 24.4 (1999), pp. 195–203.

[KT05] G. Kliewer and L. Timajev. “Relax-and-Cut for Capacitated Network Design”. In: *Proceedings of Algorithms-ESA, 2005: LNCS*. Vol. 3369. 2005, pp. 47–58.

[KV12] B. Korte and J. Vygen. *Combinatorial Optimization – Theory and Algorithms*. Springer, 2012.

[Lai+11] K. J. Lai, C. P. Gomes, M. K. Schwartz, K. S. McKelvey, D. E. Calkin, and C. A. Montgomery. “The Steiner Multigraph Problem: Wildlife Corridor Design for Multiple Species”. In: *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI ’11)*. AAAI Press, 2011. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI11/paper/view/3768.

[LD05] M. E. Lübbecke and J. Desrosiers. “Selected topics in column generation”. In: *Operations Research* 53.6 (2005), pp. 1007–1023.

[Lin+10] G. Lin, C. Nagarajan, R. Rajaraman, and D. P. Williamson. “A general approach for incremental approximation and hierarchical clustering”. In: *SIAM Journal on Computing* 39.8 (2010), pp. 3633–3669.

[Lue75] G. Lueker. *Two NP-Complete Problems in Nonnegative Integer Programming*. Tech. rep. Report No. 178, Computer Science Laboratory. Princeton University, 1975.

[Mah16] S. J. Maher. “Solving the Integrated Airline Recovery Problem Using Column-and-Row Generation”. In: *Transportation Science* 50.1 (2016), pp. 216–239. DOI: 10.1287/trsc.2014.0552.

[Mat+17] J. Matuschke, S. T. McCormick, G. Oriolo, B. Peis, and M. Skutella. “Protection of flows under targeted attacks”. In: *Operations Research Letters* 45.1 (2017), pp. 53–59. ISSN: 0167-6377. DOI: https://doi.org/10.1016/j.orl.2016.11.005.

[MBB13] İ. Muter, Ş. İ. Birbil, and K. Bülbül. “Simultaneous column-and-row generation for large-scale linear programs with column-dependent-rows”. In: *Mathematical Programming* 142.1 (2013), pp. 47–82. ISSN: 1436-4646. DOI: 10.1007/s10107-012-0561-8.

[Mer+12] G. B. Mertzios, M. Shalom, A. Voloshin, P. W. H. Wong, and S. Zaks. “Optimizing Busy Time on Parallel Machines”. In: *Proc. of the 26th IPDPS*. IEEE, 2012, pp. 238–248. DOI: 10.1109/IPDPS.2012.31.

[MMO17] J. Matuschke, S. T. McCormick, and G. Oriolo. “Rerouting flows when links fail”. In: *CoRR* abs/1704.07067 (2017). arXiv: 1704.07067. URL: http://arxiv.org/abs/1704.07067.

[MP03] R. R. Mettu and C. G. Plaxton. “The Online Median Problem”. In: *SIAM Journal on Computing* 32.3 (2003), pp. 816–832. URL: http://dx.doi.org/10.1137/S0097539701383443.

[MW84] T. Magnanti and R. Wong. “Network design and transportation planning: Models and algorithms”. In: *Transportation Sci.* 18.1 (1984), pp. 1–55.

[NP07] A. Nahapetyan and P. M. Pardalos. “A bilinear relaxation based algorithm for concave piecewise linear network flow problems”. In: *Journal of Industrial and Management Optimization* 3.1 (2007), p. 71.

[NS14] S. G. Nurre and T. C. Sharkey. “Integrated network design and scheduling problems with parallel identical machines: Complexity results and dispatching rules”. In: *Networks* 63.4 (2014), pp. 306–326.

[Nur+12] S. G. Nurre, B. Cavdaroglu, J. E. Mitchell, T. C. Sharkey, and W. A. Wallace. “Restoring infrastructure systems: An integrated network design and scheduling (INDS) problem”. In: *European Journal of Operational Research* 223.3 (2012), pp. 794–806. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2012.07.010.

[PR13] M. Poss and C. Raack. “Affine recourse for the robust network design problem: Between static and dynamic routing”. In: *Networks* 61.2 (2013), pp. 180–198. ISSN: 1097-0037. DOI: 10.1002/net.21482.

[PS95] E. W. Parsons and K. C. Sevcik. “Multiprocessor scheduling for high-variability service time distributions”. In: *Proc. of the JSSPP*. Ed. by D. G. Feitelson and L. Rudolph. Vol. 949. LNCS. Springer Berlin Heidelberg, 1995, pp. 127–145. DOI: 10.1007/3-540-60153-8_26.

[RP86] M. Richey and R. Parker. “On multiple Steiner subgraph problems”. In: *Networks* 16.4 (1986), pp. 423–438.

[RS16] A. T. Richter and S. Stiller. “Robust Strategic Route Planning in Logistics”. In: *Transportation Science* (2016).

[Sch03] A. Schrijver. *Combinatorial Optimization – Polyhedra and Efficiency*. Springer, 2003.

[SKS03] D. Simchi-Levi, P. Kaminsky, and E. Simchi-Levi. *Designing and Managing the Supply Chain: Concepts, Strategies, and Case Studies*. McGraw Hill, 2003.

[SKS10] T. Schöneberg, A. Koberstein, and L. Suhl. “An optimization model for automated selection of economic and ecologic delivery profiles in area forwarding based inbound logistics networks”. In: *Flexible services and manufacturing journal* 22.3-4 (2010), pp. 214–235.

[Sku00] M. Skutella. “Approximating the single source unsplittable min-cost flow problem”. In: *Proceedings 41st Annual Symposium on Foundations of Computer Science*. 2000, pp. 136–145. DOI: 10.1109/SFCS.2000.892073.

[SM02] A. Schulz and Martin Skutella. “Scheduling Unrelated Machines by Randomized Rounding”. In: *SIAM Journal on Discrete Mathematics* 15.4 (2002), pp. 450–469. DOI: 10.1137/S0895480199357078.

[SS13] P. Sanders and C. Schulz. “Think Locally, Act Globally: Highly Balanced Graph Partitioning”. In: *Proceedings of the 12th International Symposium on Experimental Algorithms (SEA'13)*. Vol. 7933. LNCS. Springer, 2013, pp. 164–175.

[SS14] A. J. Soper and V. A. Strusevich. “Power of Preemption on Uniform Parallel Machines”. In: *Proc. of the 17th APPROX*. Vol. 28. LIPIcs. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2014, pp. 392–402. DOI: 10.4230/LIPIcs.APPROX-RANDOM.2014.392.

[Sta03] H. Stadtler. “Multilevel Lot Sizing with Setup Times and Multiple Constrained Resources: Internally Rolling Schedules with Lot-Sizing Windows”. In: *Oper. Res.* 51.3 (2003), pp. 487–502.

[SV13] R. Sadykov and F. Vanderbeck. “Column generation for extended formulations”. In: *EURO Journal on Computational Optimization* 1.1 (2013), pp. 81–115. ISSN: 2192-4414. DOI: 10.1007/s13675-013-0009-9.

[WS11] D. P. Williamson and D. B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011, pp. I–XI, 1–504. ISBN: 978-0-521-19527-0.

[WW58] H. Wagner and T. Whitin. “Dynamic Version of the Economic Lot Size Model”. In: *Man. Sci.* 5.1 (1958), pp. 89–96.

List of Notations

General Notations

$\mathbb{R}_+$	nonnegative real numbers
$\mathbb{Z}_+$	nonnegative integer numbers
$\mathbb{N}$	natural numbers
$\mathcal{NP}$	nondeterministic polytime complexity class
$\mathrm{conv}(A)$	convex hull of the set A
E	set of edges in the network
$\mathrm{start}(e)$	start node (head) of edge e
$\mathrm{end}(e)$	end node (tail) of edge e
$\delta^+(n)$	set of outgoing edges of node n
$\delta^-(n)$	set of incoming edges of node n
$\delta(n)$	incident edges to node n for undirected graphs, or incoming and outgoing edges for directed graphs
min, max	minimum or maximum, or infinmum or supremum, if taken over the empty set
$u \rightsquigarrow v$	a representative from the set of shortest paths from node u to v w.r.t. some cost function
$(a)^+$	short for $\max\{0, a\}$

Consolidation Flows

K	set of physical commodities
Π	set of relevant properties
J	set of all demands
$\alpha_{j\pi}$	extent of property π for demand j
source_j	source node for demand j
sink_j	sink node for demand j
$D_j = (\mathrm{source}_j, \mathrm{sink}_j, \alpha_j)$	tuple of source node, sink node and property vector for demand j
$\mathcal{P}, \mathcal{P}_j$	set of all possible source-sink paths, resp. all paths to satisfy demand j
$j(v, \ell, S)$	join function: the minimum cost for an in-tree routing of $S \subseteq J$ to v with bounded height ℓ
$s(v, \ell, S)$	split function: the minimum cost for an out-tree routing from $S \subseteq J$ out of v with bounded height ℓ
$\tilde{j}(v, \ell, S)$	pre join function
$\tilde{s}(v, \ell, S)$	pre split function

Deterministic Model

$K, K(e)$	set of tariff levels, resp. those available for edge e
g_k	cost for container/segment k

$\beta_{\pi k}$	capacities of tariff level k in property π
f_k	variable: number of copies to buy of tariff level k
c_e^T	edge cost for edge e arising with exact tariff selection
$\bar{J}$	set of all super demands
$J(j)$	set of original demands that are aggregated into super demand j
d_j	requested transport units for demand j
$\varrho_{j\pi}$	extend of property π for aggregated demand j
r_j^P	assign demand j to path P
c_j^P	cost of assigning demand j to path P

Robust Model

$d_j \in [\bar{d}_j, \hat{d}_j]$	requested units of demand j are uncertain within this interval
$\tilde{d}_j$	possible additional units: $\tilde{d}_j = \hat{d}_j - \bar{d}_j$
$\mathcal{R}$	a path solution to a routing problem
Γ	robustness budget: number of demands that may deviate simultaneously in a worst case scenario
c_e	variable: adversary cost for edge e
μ_j	decision: let demand j deviate in a worst case scenario
$h_{k\pi}$	decision: let property π be cost driving in tariff level k
$h'_{kj\pi}$	decision: let property π of j deviate for tariff level k in a worst case
$\rho_{j\pi}$	decision: let property π of j deviate in a worst case

Obtaining a Solvable Model

y_k	decison: open the facility associated with tariff level k
o_k	cost for opening facility k
$\ell_k(d)$	function: maximum property usage for facility k
ℓ_k	variable: maximum property usage for facility k
z_{kj}	decision: assign demand j to facility k
m_k	linear cost factor for facility k
c_k	function: partially linearized cost for tariff level k
$c_e^L(d)$	function: simplified cost for edge e and demand values d
$\sigma_{kj\pi}$	capacity usage of demand j of facility k w.r.t property π, $\sigma_{kj\pi} := \alpha_{j\pi}/\beta_{\pi k}$
Z	a facility assignment solution
λ_k	variable: adversary's maximum property usage for fac k
$s_{k\pi}$	slack variables for Inequalities 4.30
$\theta_{k\pi}$	decision: wich property π of facility k is tight in the adversary problem
$\tilde{\sigma}^*_{kj\pi}$	$= \sigma_{kj\pi}\tilde{d}_j z_{kj}$
δ_j	dual variables for Inequalities (4.38)
ϕ_k	dual variables Inequalities (4.35)
γ_{kj}	dual variables Inequalities (4.39)
$\xi_{kj\pi}$	dual variables Inequalities (4.40)
$D_j(e)$	upper bound of possible flow for demand j on edge e
$a(d)$	aggregated property consumption for demand vector d
$c_e^V(j,d)$	edge cost share for demand j when d is routed along e
$c^{\text{DIR}}(j)$	exact cost of a cheapest path from source$_j$ to sink$_j$
$t_\pi(e)$	tightness factor for property π for tariff selection on edge e
$\mathcal{U}(P)$	the set of facility paths for path P

G_j	aggregated robustness budget for super demand j
$A_j(\Gamma_j)$	set of possible property vectors for an uncertain super demand j
$B_j(\Gamma_j)$	relaxed version of set $A_j(\Gamma_j)$ see (4.53)

Names for Problem Formulations

ROUTE	a MILP for the routing problem with exact tariff cost
ADV	the adversary problem for ROUTE with exact tariff cost
ROBROUTE	robust counterpart for ROUTE
HUBROUTE	the deterministic k-median hub location problem
HUBROBROUTE	robust counterpart for HUBROUTE
ROUTELIN	the version of ROUTE with approximate tariff cost
ADVLIN	the version of adversary problem ADV for approximate tariff cost
ADVLIN2	a solvable model for the adversary obtained by linearizing ADVLIN
ADVDUAL	the dual of ADVLIN2
ROB-LIN	a solvable model for ROBROUTE with approximate tariff cost and relaxed Adversary
HUBLINPATH	a linearization of problem HUBROUTE
HUBLINPATHDUAL	the dual of HUBLINPATH
HUBLINEDGE	an arc node formulation of HUBLINPATH
HUBLINEDGEDUAL	the dual of HUBLINEDGE
ROB-HUBLINPATH	a solvable model for HUBROBROUTE
ROB-HUBLINPATHDUAL	the dual of ROB-HUBLINPATH
VIOLATION	the violation lp for column-and-row generation
ROB-VIOLATION	the variant VIOLATION for HUBROBROUTE

Symbols for Hub Location

$n \in N$	a hub node from the set of all hub nodes
o_n	fix cost for opening a hub node
g_n	decision: open hub node n
M	the maximum admissable number of hubs to open
κ	variable: dual hub costs for (5.10)
RZ_{ej}	variable: dual edge cost variable for (4.27)
$ZL_{k\pi}$	variable: linear property price on facility k for property π in the current pricing LP (could be fixed to the prices in the master LP) (4.28)
R_j	variable: dual path price for (4.4)
ZY_{kj}	variable: dual facility cost share for (4.24)
YG_{ne}	variable: dual hub cost share for (5.9)
$\tilde{\varrho}_{j\pi}$	possible additional deviation (boxed uncertainty set version) of property π for aggregated demand j

LP Sased Search

I_n	variable: violation at hub n
SL_{ekj}	variable: slack of constraints (4.30)
$hand_j(n), hand_j(e)$	handling cost at node n or accumulated into edge e for demand j
B_{jn}	variable: distance label of potential shortest path tree
$x_j(e)$	variable: flow value of demand j on edge e

B_{jn}	variable: the distance label of a shortest path tree (rooted at sink sink_j) wrt. dual edge costs RZ_{ej}
$\bar{J}(e), \bar{J}(k)$	the set of demands that is already present in restricted master LP
F_j	the set of nodes $n \in F_j \subseteq N$ for which the potential B_{jn} is considered fixed in VIOLATION

Aggregation Techniques

$\mathcal{O}$	set of source nodes
$\mathcal{D}$	set of sink nodes
$\hat{\mathcal{O}}$	set of meta–sources
$\hat{\mathcal{D}}$	set of meta–sinks
ψ	function: assignes meta–nodes to demands
c_{ij}	assignment cost for assigning client j to facility i
o	opening cost for opening facility i
$\eta_{kj\pi}$	proportion of facility k used by the extend of property π for super demand j
$\tilde{\eta}_{kj\pi}$	proportion of facility k used by the extend of property π for deviating super demand j
T	The set ob observations
t	index for one observation